FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Protótipo de testes para aprendizagem com classificadores múltiplos

Ana Maria Lima Fernandes

Mestrado Integrado em Engenharia Informática e Computação

Orientador: João Pedro Carvalho Leal Mendes Moreira (Prof. Doutor)

17 de Janeiro de 2011

Protótipo de testes para aprendizagem com classificadores múltiplos

Ana Maria Lima Fernandes

Mestrado Integrado em Engenharia Informática e Computação

Aprovado em provas públicas pelo Júri:

Presidente: Rosaldo José Fernandes Rossetti (Doutor)

Vogal Externo: José Manuel Matos Moreira (Doutor)

Orientador: João Pedro Carvalho Leal Mendes Moreira (Doutor)

_____

11 de Fevereiro de 2011

Resumo

Os classificadores múltiplos são processos que utilizam um conjunto de modelos, cada um deles obtido pela aplicação de um processo de aprendizagem para um problema dado. Combinam vários classificadores individuais, em que para cada um deles são utilizados dados de treino para gerar limites de decisão diferentes. As decisões produzidas pelos classificadores individuais contém erros, que são combinados pelos classificadores múltiplos de forma a reduzir o erro total.

Estes têm vindo a ganhar uma crescente importância devido principalmente ao facto de permitirem obter um melhor desempenho quando comparado com o obtido por qualquer um dos modelos que o compõem, principalmente quando as correlações entre os erros cometidos pelos modelos de base são baixos. A investigação nesta área tem crescido, tornando-se uma área de investigação importante.

No entanto, para que o desempenho seja melhor do que o desempenho obtido por cada classificador individualmente, é necessário que cada um deles produza uma decisão diferente originando uma diversidade de classificação. Esta diversidade pode ser obtida tanto pela utilização de diferentes conjuntos de dados para o treino individual de cada classificador, como também pela utilização de diferentes parâmetros de formação de diferentes classificadores.

Apesar disso, a utilização de classificadores múltiplos para aplicações no mundo real pode apresentar-se como dispendiosa e morosa.

Tem-se notado nos dias de hoje que o desenvolvimento web tem vindo a crescer exponencialmente, assim como o uso de bases de dados. Desta forma, combinando a forte utilização da linguagem R para cálculos estatísticos com a crescente utilização das tecnologias web, foi implementado um protótipo que facilitasse a utilização dos classificadores múltiplos, mais precisamente, foi desenvolvida uma aplicação web que permitisse o teste para aprendizagem com classificadores múltiplos, sendo utilizadas as tecnologias PHP, R e MySQL. Com esta aplicação pretende-se que seja possível testar algoritmos independentes do software em que estejam desenvolvidos, não sendo necessariamente escritos em R.

Nesta Dissertação foi utilizada a expressão “classificadores múltiplos” por ser a mais comum, apesar de ser redutora e existirem outros termos mais genéricos como por exemplo modelos múltiplos e *ensemble learning*.

Palavras-chave: classificadores múltiplos, geração de modelos, selecção de modelos, combinação final.

Abstract

Ensemble learning is a process that uses a set of models, each obtained by applying learning process for a given problem. It combine several individual classifiers, where for each classifier are used training data to generate different decision limits. The decision made by individual classifier contains errors, which are combined by the ensemble learning to reduce the total error.

These have been gaining an increasing importance mainly because they allow better performance when compared with those obtained for any of models that comprise, especially when the correlations between the errors committed by the base models are low. Research in this area has grown, becoming an important research area.

However, that performance is better than the performance achieved by each classifier individually, it is necessary that each produces a different decision causing a variety of classification. This diversity can be obtained either using different data sets for the individual training of each classifier, as well by using different parameters of formation of different classifier.

Nevertheless, the use of the ensemble learning for applications in the real world can present as costly and time consuming.

Nowadays, the web development has grown exponentially as well as the use of databases. Thus, combining the strong use of the r language for statistical calculation with the increasing use of web technologies, we built a prototype that would facilitate the use of ensemble learning, more precisely, we developed a web application that allows the testing for ensemble learning, being used PHP, MySQL and R language. With this application is expected to be able to test algorithms, regardless the software as they are developed, not necessarily written in R.

Keywords: Ensemble learning, ensemble generation, ensemble pruning, ensemble integration.

À minha mãe

Agradecimentos

Ao chegar ao final desta Dissertação não podia deixar de expressar os meus sinceros agradecimentos às pessoas que de uma forma ou outra contribuíram para que fosse possível chegar aqui hoje, com destaque para o meu orientador Prof. Dr. João Mendes Moreira pelo seu empenho e disponibilidade no acompanhamento desta Dissertação.

À minha família pelo apoio incondicional que sempre demonstrou.

Ao professor Raul Moreira Vidal pela sua disponibilidade.

Ao meu colega Girson Monteiro pela disponibilidade e apoio incondicionais que demonstrou.

Aos meus amigos e colegas pelo apoio e incentivo demonstrados durante este tempo e pelo companheirismo.

Finalmente à todas as pessoas que de uma forma ou outra contribuíram para que fosse possível chegar aqui. Os meus sinceros agradecimentos.

Ana Maria Lima Fernandes

Índice

Lista de Figuras

Lista de Tabelas

Abreviaturas e Símbolos

HTML	Hyper Text Markup Language
KNN	k-nearest neighbors
PHP	Hypertext Preprocessor
PPR	Projection Pursuit Regression
RF	Random Forest
SAS	Statistical Analysis Software
SPSS	Statistical Package for the Social Sciences
SQL	Structured Query Language
SVM	Support Vector Machines
WEKA	Waikato Environment for Knowledge Analysis

1 Introdução

1.1 Contexto/Enquadramento

Nos últimos tempos tem-se intensificado a utilização de algoritmos no sentido de interagir com sistemas inteligentes. Uma das técnicas utilizadas neste sentido refere-se aos classificadores múltiplos que são sistemas constituídos por vários classificadores individuais e de preferência diferentes que produzem resultados singulares, que com uma combinação estratégica produzem um resultado final único para o sistema. Este processo começa com a aplicação de algoritmos iniciais que produzem vários modelos levando a uma integração posterior que finalmente produzirá um resultado único para todo o processo. O processo experimental é complexo, podendo ser moroso e os resultados apresentarem uma certa dificuldade na interpretabilidade.

1.2 Descrição do problema

Os classificadores múltiplos são constituídos por vários classificadores singulares, em que cada um deles produz uma decisão que será combinada no final do processo para que possa ser produzido um resultado final único com melhor desempenho do que os resultados produzidos por cada classificador individual. No entanto é necessário haver uma certa diversidade entre os vários classificadores para que se possa produzir um resultado final com maior precisão, o que pode ser obtido pela utilização de diferentes conjuntos de dados, assim como diferentes algoritmos de indução. Tudo isso leva a que a quantidade de dados seja ainda maior e que o processo experimental por sua vez seja mais complexo traduzindo-se numa maior dificuldade na interpretabilidade. Isso porque os parâmetros a serem utilizados no processo são variáveis e os dados além de serem em grandes quantidades também podem e devem variar. Desta forma e para que o processo seja o mais simples possível ao utilizador, foi desenvolvido um protótipo a

ser usado em aprendizagem com múltiplos classificadores, mais precisamente, consiste no desenvolvimento de uma aplicação Web que facilite os testes de algoritmos de previsão. Desta forma, pretende-se que seja possível testar algoritmos em linguagens diferentes, podendo ser usados tanto por utilizadores comuns, através da utilização de código já desenvolvido ou por especialistas que escrevem o seu próprio código para testar.

1.3 Motivação e Objectivos

Os classificadores múltiplos estudam algoritmos e arquitecturas de forma a construir modelos ou conjuntos de classificadores. Estes sistemas apresentam resultados favoráveis com uma maior precisão quando comparados com sistemas com único classificador [Pol06]. A sua utilização tem vindo a intensificar-se e as pesquisas nesta área têm-se expandido rapidamente, tornando-se assim um importante tema de pesquisa.

Apesar de apresentar bons resultados, podem haver problemas de interpretabilidade e dificuldades nos problemas heterogéneos porque apresentam um processo experimental complexo. Uma das motivações para a realização deste trabalho é que se possa desenvolver uma aplicação que facilite o processo experimental dos classificadores múltiplos.

Sendo assim os objectivos definidos para a elaboração deste projecto passam principalmente pela construção de uma aplicação que facilite a componente de testes para algoritmos de previsão, pois os conjuntos de aprendizagem existem, apresentam bons resultados, mas muitas vezes existem problemas de interpretabilidade e dificuldades no processo experimental. Ainda podemos destacar o facto de se poder utilizar algoritmos desenvolvidos em diferentes aplicações com por exemplo: R, WEKA, RapidMiner, etc e a possibilidade de poder ser usado tanto por especialista fazendo código para as diferentes fases, como por utilizadores comuns utilizando código existente.

1.4 Estrutura da Dissertação

Esta tese encontra-se estruturada em seis capítulos, sendo eles:

No Capítulo 1 foi feita uma Introdução onde foram apresentadas as motivações e feita uma descrição do problema que levou a que fossem traçados os objectivos.

No Capítulo 2, intitulado “Classificadores múltiplos” fala-se sobre os classificadores múltiplos, as suas fases e ainda são apresentados alguns métodos utilizados nos classificadores múltiplos.

No Capítulo 3 são descritas as tecnologias utilizadas no desenvolvimento do referido trabalho, assim como uma revisão bibliográfica sobre as ferramentas que trabalham com classificadores múltiplos.

O capítulo 4 fala sobre o funcionamento da aplicação desenvolvida assim como os detalhes referentes à implementação do trabalho.

No capítulo 5 são apresentadas algumas experiências realizadas com a aplicação desenvolvida.

Finalmente, no Capítulo 6 são apresentadas as conclusões retiradas do trabalho realizado e trabalho futuro.

2 Classificadores múltiplos

Neste capítulo pretende-se descrever de forma pormenorizada os sistemas de classificadores múltiplos, bem como as fases que o constituem. São ainda apresentados alguns métodos utilizados em classificadores múltiplos, sendo que para problemas diferentes alguns métodos são mais adequados do que outros. Finalmente é apresentado um resumo sobre o tema em questão.

2.1 Introdução

Classificadores múltiplos são processos que utilizam um conjunto de modelos, cada um deles obtido pela aplicação de um processo de aprendizagem para um problema dado. Este conjunto de modelos é integrado de alguma forma para obter a previsão final [Mor08]. Construir um conjunto de classificadores é uma forma comum de melhorar o desempenho do modelo resultante para tarefas de classificação e regressão [WCTS07]. Dado que este processo é constituído por N classificadores diferentes, são fornecidos os dados de entrada, que produzirão N resultados individuais e estes serão combinados numa saída única para o problema. A figura seguinte representa o esquema geral dos classificadores múltiplos, em que dada uma determinada entrada, existem diversos classificadores que serão combinados de uma determinada forma e produzirão uma única saída.

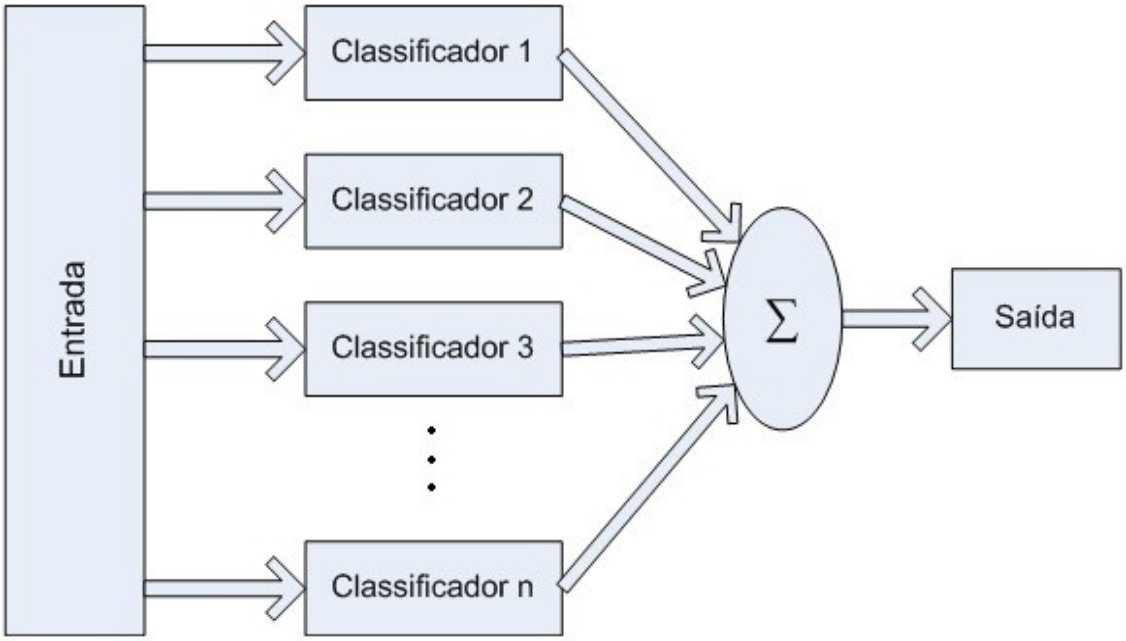

Figura 2.1: Esquema de classificadores múltiplos

Ao utilizar os classificadores múltiplos, pretende-se resolver um problema de inteligência computacional, onde são utilizados diversos modelos para obtenção de uma previsão dos modelos como um todo. Esta previsão é melhor do que as previsões obtidas por cada um dos modelos individualmente. A seguir é ilustrado um exemplo de como um classificador múltiplo, formado por vários classificadores individuais produz um melhor resultado do que os classificadores individuais.

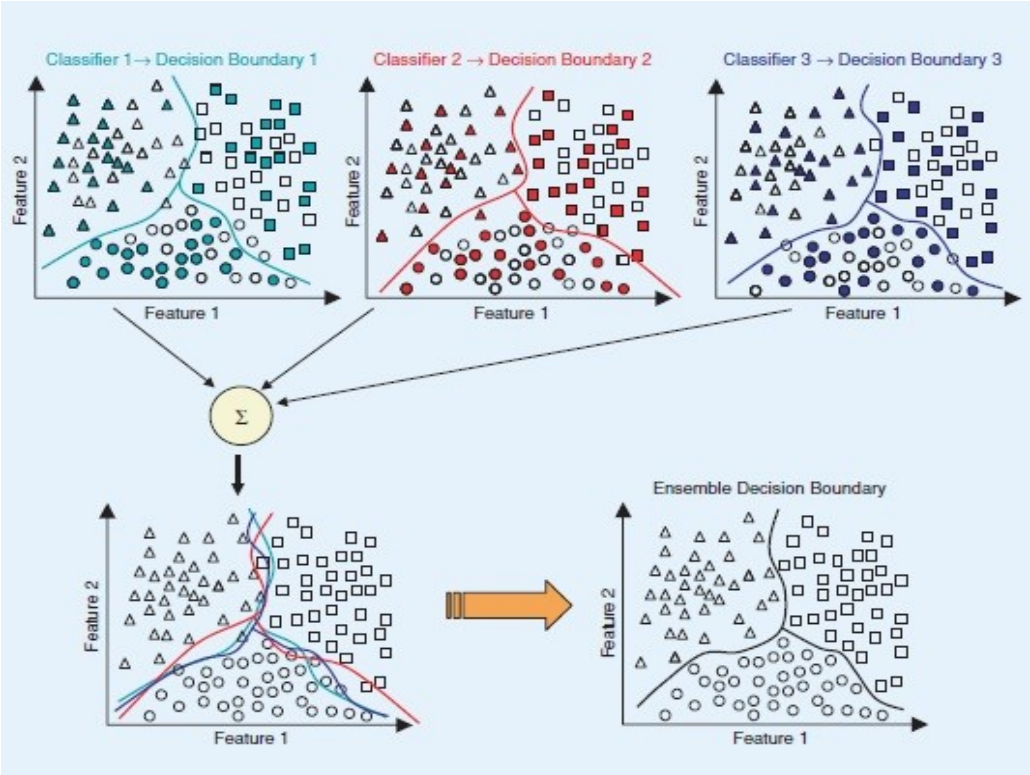

Figura 2.2: Exemplo de um resultado produzido por um classificador múltiplo [Pol06]

Na figura acima, o conjunto é formado por três classificadores, em que cada um deles produz um resultado diferente com determinado erro. É claramente visível que depois da combinação o resultado final é melhor do que qualquer um dos resultados individuais de cada classificador.

Os classificadores múltiplos têm vindo a ganhar maior destaque e muitos utilizadores preferem utilizá-los em vez de um classificador individual pelas seguintes razões:

- razões estáticas - produzem um melhor resultado quando comparado com os classificadores individuais, apresentando uma melhor precisão [Pol06];
- existência de grandes quantidades de dados - em certas aplicações, a quantidade de dados é muito grande para ser tratada eficazmente por um único classificador [Pol06]. Desta forma é preferível fazer uma partição dos dados em subconjuntos menores e assim classificadores diferentes com conjuntos de dados, produzirão resultados diferentes. Resultados esses que serão combinados utilizando uma determinada regra de combinação estratégica de forma a produzir um melhor resultado.
- Pequenas quantidade de dados - quando a quantidade de dados é pequena os classificadores múltiplos podem também ser usados [Pol06] garantindo desta forma a disponibilidade de dados necessários ao treino.
- Dividir e conquistar – os problemas, independentemente da quantidade de dados que possuam, nem sempre são fáceis de resolver quando se tem apenas um modelo no processo de resolução. Neste caso, uma solução seria dividir o conjunto de dados em vários subconjuntos, dependendo do número de divisões necessárias para tornar a resolução do problema mais fácil. Desta forma, diz-se que o sistema segue uma abordagem "*divide-and-conquer*" pela divisão do espaço de dados em partições pequenas e mais simples, onde cada modelo aprende apenas uma das partições simples [Pol06]. As duas figuras seguintes ilustram a abordagem "*divide-and-conquer*", em que a primeira ilustra uma tomada de decisão considerada complexa para um único modelo e a segunda mostra que o mesmo espaço pode ser dividido em pequenos sub-espaços e assim tomar uma melhor decisão;

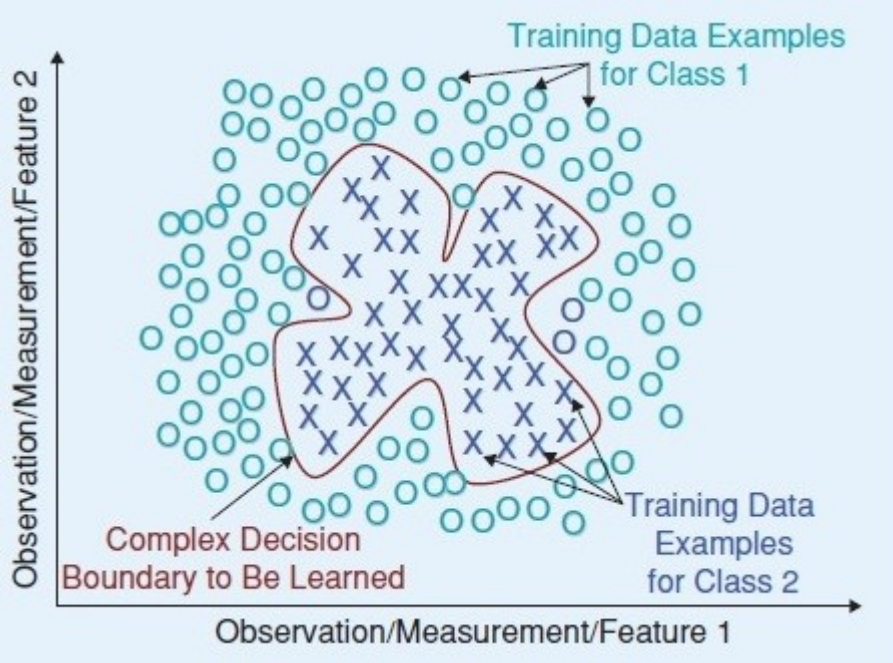

Figura 2.3: Fronteira de decisão complexa para um único classificador [Pol06]

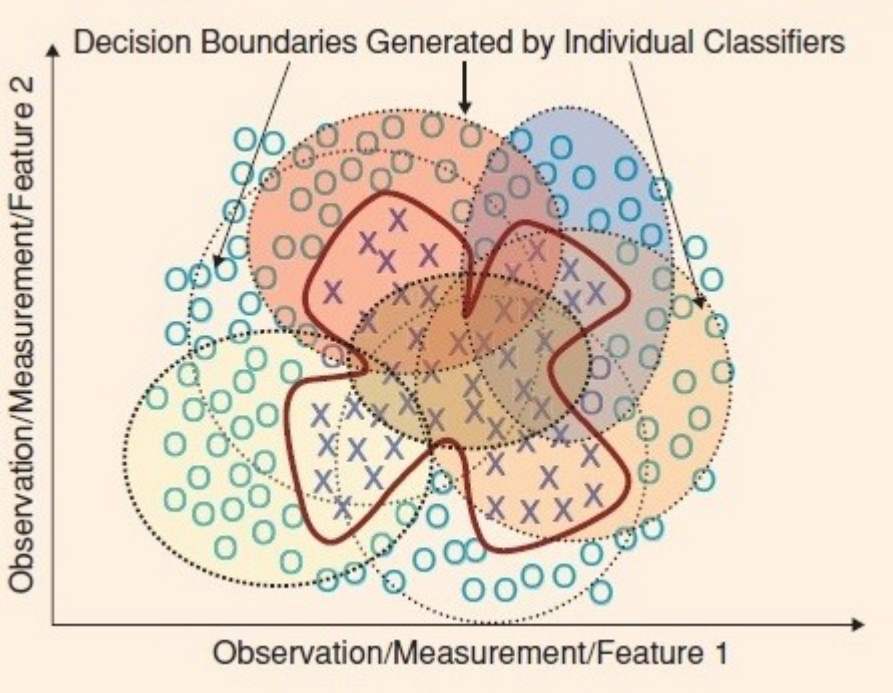

Figura 2.4: Espaço de dados dividido por vários modelos em um conjunto [Pol06]

- fusão de dados [Pol06] – quando existem vários conjuntos de dados, provenientes de diversas fontes com características diferentes, não se pode usar um classificador

único para aprender toda a informação existente, pelo que em vez disso, devem ser utilizados os classificadores múltiplos.

O processo de classificadores múltiplos encontra-se dividido em 3 fases, sendo elas: a geração do conjunto de modelos a partir de algoritmos de indução, selecção de modelos através da aplicação de algoritmos de selecção e integração final que é a combinação dos resultados obtidos por cada um dos modelos de modo a obter de um resultado único para o conjunto, embora alguns autores considerem apenas a geração do conjunto de modelos e a integração final [RPAT04]. A figura a seguir ilustra as três fases do processo de classificadores múltiplos:

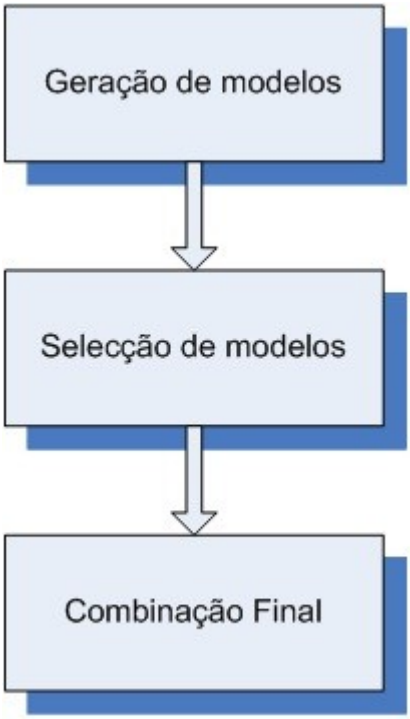

Figura 2.5: Fases do processo de classificadores múltiplos

2.1.1 Geração de modelos

A geração de modelos é a primeira fase de classificadores múltiplos e consiste na geração de um conjunto de modelos. Estes modelos são gerados pela utilização de algoritmos de indução. Caso seja utilizado um único algoritmo, os modelos são chamados homogéneos, caso sejam gerados por algoritmos diferentes são designados de modelos heterogéneos. Os modelos heterogéneos garantem que haja diversidade, pelo facto de serem gerados através da utilização de mais do que um algoritmo de indução, o que pode dar melhores resultados. A diversidade nos modelos homogéneos pode ser obtida de formas diferentes: através da manipulação dos

dados utilizados no treino dos modelos ou através da manipulação do processo de geração do modelo.

No primeiro caso, isto é, na geração de modelos diferentes através da manipulação de dados, existem tipicamente três formas de o fazer:

- Sub-amostragem do conjunto de treino – os modelos são obtidos usando diferentes sub-amostras do conjunto de treino. Assume-se que os algoritmos são instáveis, isto é, pequenas perturbações nos dados de treino originam modelos diferentes. Como exemplos deste tipo de algoritmos temos as árvores de decisão e as redes neuronais.
- Manipulação de dados de entrada – neste tipo de manipulação a obtenção de diferentes conjuntos é conseguida pela alteração da representação de exemplo. Novos conjuntos são gerados pela substituição de um conjunto original, existindo duas abordagens diferentes: selecção de um subconjunto original de atributos e aplicação de transformações sobre os atributos originais.
- Manipulação de dados de saída – nesta abordagem ainda não existem muitas pesquisas e as que existem centram-se maioritariamente na classificação.

No segundo caso, isto é, obtenção da diversidade através do processo de geração de modelos, pode ser feita de três formas diferentes:

- manipulação do conjunto de parâmetros – nesta abordagem alterações nos parâmetros originam diversidade, pelo que os algoritmos de indução utilizados devem ser sensíveis aos parâmetros. O grau de sensibilidade varia de parâmetro para parâmetro. Deve-se dar maior importância aos parâmetros aos quais o algoritmo é mais sensível de forma a que a diversidade seja maior possível.
- manipulação do algoritmo de indução – a diversidade neste caso é conseguida através de alterações no próprio algoritmo de indução, levando a que o mesmo algoritmo produza resultados diferentes para o mesmo conjunto de dados.
- manipulação do modelo – a diversidade é conseguida a partir de um único modelo. Isto é, tendo um modelo, é aplicado um conjunto de regras repetidamente para obter um conjunto de modelos.

2.1.1.1 Métodos utilizados na geração de modelos

Existem vários métodos utilizados na geração de modelos, alguns deles apropriados aos modelos homogéneos, outros aos modelos heterogéneos. Os métodos como: *Bagging* e *Boostin*g são usados na geração de modelos homogéneos, enquanto que S*tacked Generalization* é utilizada na geração de modelos heterogéneos.

Bagging

O termo *bagging* foi introduzido por Breiman como um acrónimo para "*Bootstrap AGGregatING*" [Kun04]. Foi um dos primeiros métodos eficazes para classificadores múltiplos [Sew07]. Apesar de ter sido concebido para a classificação, pode ser usado tanto para classificação como para regressão, em que para a classificação, as saídas dos modelos são combinadas pelo voto e para o caso de regressão são combinadas pela média. Neste método, a diversidade necessária para fazer funcionar o conjunto é conseguida usando diferentes conjuntos de treino. É um método que funciona bem para classificadores instáveis [Brei96], isto é, qualquer alteração no conjunto de treino pode causar uma mudança significativa no modelo. Se os classificadores usados não forem instáveis, o resultado produzido será um conjunto que não apresenta diversidade, pelo que não melhorará o desempenho global. Redes neuronais e árvores de decisão são exemplos de classificadores instáveis.

Boosting

O algoritmo *boosting* criado em 1990 por Schapire, foi inspirado por um algoritmo de aprendizagem online chamado Hedge(β) [Kun04], que atribui pesos a um conjunto de estratégias utilizadas para prever o resultado de um determinado evento. As estratégias com previsão correcta têm peso maior, enquanto que as com previsão incorrecta têm peso reduzido. *Boosting* é um meta-algoritmo que pode ser visto como um método de média de modelo. É o método de classificadores mais utilizados e uma das ideias mais poderosas de aprendizagem introduzida nos últimos 20 anos [Sew07]. Foi originalmente concebido para a classificação, podendo ser estendido para regressão.

O *boosting* inclui o método *AdaBoost* em que o nome é uma abreviação de "*ADAptative BOOSTing*". É o algoritmo de *boosting* mais popular [Sew07]. Utiliza o mesmo conjunto de treino mais do que uma vez e pode combinar um número arbitrário de algoritmos bases, adicionando um classificador de cada vez. Uma das razões que explica o sucesso deste método é o facto de conduzir o erro de treino do conjunto para zero rapidamente, logo nas primeiras iterações [Kun04].

Stacked Generalization

Stacked generalization desenvolvido por Wolpert em 1992, é uma forma diferente de combinar vários modelos, que introduz o conceito de meta-algoritmo [Sew07]. Com este método pretende-se reduzir a taxa de erro de generalização de um ou mais generalizadores. Quando usado com vários generalizadores, a generalização empilhada pode ser vista como uma versão mais sofisticada da validação cruzada e ao ser utilizada com um único generalizador, pode ser vista como um regime de estimativa do erro de um generalizador que tenha sido treinado em um conjunto de aprendizagem.

No trabalho em questão, deu-se uma maior importância aos modelos heterogéneos e, para tal, destacou-se este método para a geração de modelos. Neste método, a aprendizagem é feita em dois níveis: um primeiro nível constituído pelos dados iniciais e um segundo formado pelos dados de treino que são as previsões dos classificadores de base obtidos através da validação cruzada (sendo que a validação cruzada é feita em duas etapas: treino e teste) e os dados de teste que são as previsões dos classificadores de base no conjunto de teste. As figuras seguintes ilustram as etapas de treino e teste, em que na fase de treino existem 4 algoritmos que produzem 4 modelos (cada um produzido por um algoritmo) e na fase de teste os 4 modelos serão combinados originando um meta modelo.

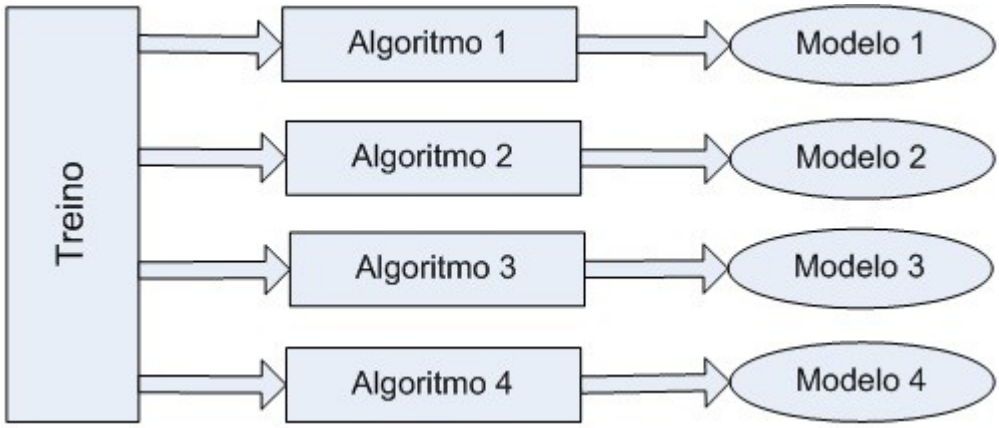

Figura 2.6: Fase de treino da validação cruzada

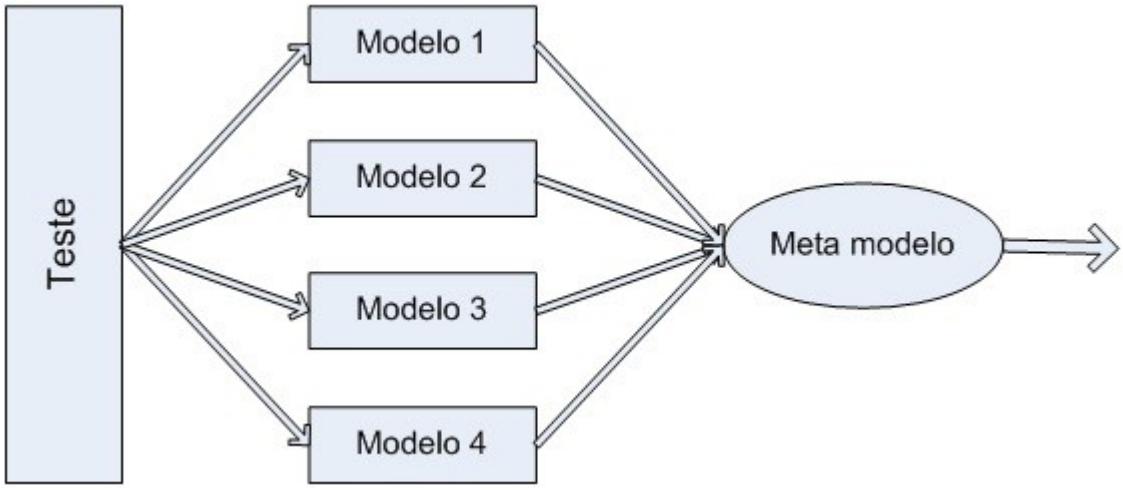

Figura 2.7: Fase de teste da validação cruzada

Random Subspace Method

Random Subspace Method é o método em que máquinas de aprendizagem são treinadas em sub-espaços escolhidos aleatoriamente do espaço de entrada original. Este é um método relativamente recente de combinar modelos [Sew07]. Este método geralmente é utilizado para classificadores lineares, *svm* (support vector machine) e outros tipos de classificadores e a saída dos modelos são combinados, geralmente por votação maioritária.

Em resumo não se pode afirmar que um determinado método de geração de modelos é melhor do que outro. O que se pode dizer é que dependendo do problema ser de regressão ou classificação alguns métodos são mais adequados e mais promissores relativamente à outros. No geral os métodos existentes apresentam resultados bastante satisfatórios quando comparados com algoritmos individuais, garantindo melhor desempenho e proporcionando diversidade nos vários modelos a serem usados nas fases seguintes.

A geração de modelos pode ser demorada se a quantidade de dados a serem utilizados for grande e se o número de modelos a serem gerados também for grande.

2.1.2 Selecção de modelos

A segunda fase do processo de classificadores múltiplos, isto é, a selecção é o processo de selecção de um subconjunto de modelos a partir de um conjunto produzido anteriormente de forma a melhorar a capacidade de previsão ou ainda reduzir o custo computacional [Mor08]. Se o número de modelos for muito grande, a selecção pode tornar-se exaustiva. Apesar de alguns autores não considerarem a fase de selecção de modelos no processo de classificadores múltiplos, ela mostra-se importante pelas seguintes razões:

- evita a combinação de modelos muito semelhantes, garantindo assim a diversidade dos modelos pertencentes ao resultado;
- modelos com baixa eficiência podem deteriorar a saída dos classificadores múltiplos;
- classificadores múltiplos constituídos por grande número de elementos não acrescentam melhorias significativas relativamente à classificadores múltiplos formado por um número eficiente de elementos, podendo mesmo piorar o seu desempenho além de aumentar desnecessariamente a complexidade.

Muitos dos métodos utilizados na selecção de modelos têm em conta o grau de diversidade, uma vez que um bom resultado não é garantido apenas pela precisão dos modelos base, mas também pela diversidade do conjunto. Portanto os modelos escolhidos para além de serem os mais eficientes, são também os que apresentam uma maior diversidade.

A selecção dos modelos pode ser classificada de acordo com o algoritmo de busca utilizado, podendo ser: sequencial, aleatória ou exponencial, selecção por ordenação e algoritmos aglomerados.

Algoritmos de selecção sequencial

Estes algoritmos alteram uma solução iterativamente adicionando ou removendo modelos e utiliza três tipos de algoritmos de busca:

Selecção para a frente - a pesquisa começa com um conjunto vazio a vai adicionando modelos ao conjunto, procurando diminuir o erro de previsão esperado.

Selecção para trás – quando a pesquisa começa com todos os modelos do conjunto e vai eliminando modelos em cada iteração de forma a reduzir o erro de previsão esperado.

Selecção para a frente e para trás – a selecção é constituída tanto por passos de selecção para a frente como para trás. É utilizada com o objectivo de evitar situações em que melhorias rápidas no início das iterações não permitem soluções com melhorias menores no início, mas com melhor resultado final.

Algoritmos de selecção aleatórios

Para estes algoritmos são atribuídos pesos aleatórios a cada um dos modelos base, aplicando depois um algoritmo genético de forma a caracterizar o contributo do respectivo modelo para o conjunto.

Algoritmos de selecção exponenciais

Os algoritmos exponenciais referem-se a situações em que a selecção do subconjunto é efectuada de forma exaustiva. Se se quer seleccionar modelos de um conjunto com K modelos, o espaço de pesquisa possui 2^K-1 subconjuntos não vazios, pelo que deve ser utilizada quando existem poucos modelos, pois para quantidade superior a 30, torna-se um problema intratável.

Algoritmos de selecção por ordenação

Estes algoritmos classificam os modelos de acordo com um determinado critério e geram um conjunto contendo os *k* modelos melhores classificados do ranking. O valor desse k é

determinado de acordo com um dado critério, critério este que pode ser muito variado, dependendo do objectivo final da classificação. Desta forma são seleccionados os melhores k modelos de um vasto conjunto. No entanto, apesar de serem seleccionados os melhores modelos, não é garantida a diversidade do conjunto seleccionado, pelo que o resultado pode não ser bom. Sendo assim uma forma de garantir diversidade e consequentemente bons resultados é utilizando conjuntos heterogéneos e garantir que o critério de selecção privilegie a diversidade.

Algoritmos aglomerados

Nestes algoritmos os modelos são agrupados em vários grupos onde é escolhido um ou mais modelos representativos de cada grupos [Mor08]. Os modelos são testados pela sua precisão, começando por aqueles que apresentem uma menor precisão. Estes vão sendo eliminados e o modelo que apresentar maior precisão é escolhido como sendo representativo do grupo.

Na selecção de modelos, os vários algoritmos apresentam resultados diferentes para os diferentes problemas, sendo que alguns algoritmos são mais apropriados para certos problemas enquanto que outros se adequam mais a outros problemas.

2.1.3 Integração Final

A integração final consiste na combinação dos modelos seleccionados na fase de selecção, de uma forma estratégica de modo a produzir uma resposta ao problema e também diminuir o erro total, melhorando a previsão. É a terceira e última fase do processo de classificadores múltiplos: inicialmente foram gerados um conjunto de modelos com certas diferenças entre eles. A seguir são aplicados certos algoritmos de selecção de forma a seleccionar os melhores modelos do conjunto. Mas não basta ter, de entre o vasto conjunto de modelos iniciais, aqueles que fazem parte do resultado final. O resultado final é obtido combinando os modelos seleccionados anteriormente. Só assim é possível obter um melhor desempenho dos classificadores múltiplos. Esta combinação utiliza as previsões feitas pelos modelos no conjunto para obter a previsão do conjunto final, e dependendo da tarefa, será aplicado um método de combinação específico. Os métodos utilizados na combinação podem ser divididos em dois grupos:

- Algébricos onde encontramos: média, média ponderada, soma, soma ponderada, produto, máximo, mínimo, mediana.
- Métodos baseados em votação, onde podemos destacar: votação maioritária e votação maioritária ponderada.

A figura seguinte ilustra alguns destes métodos que são usados na classificação.

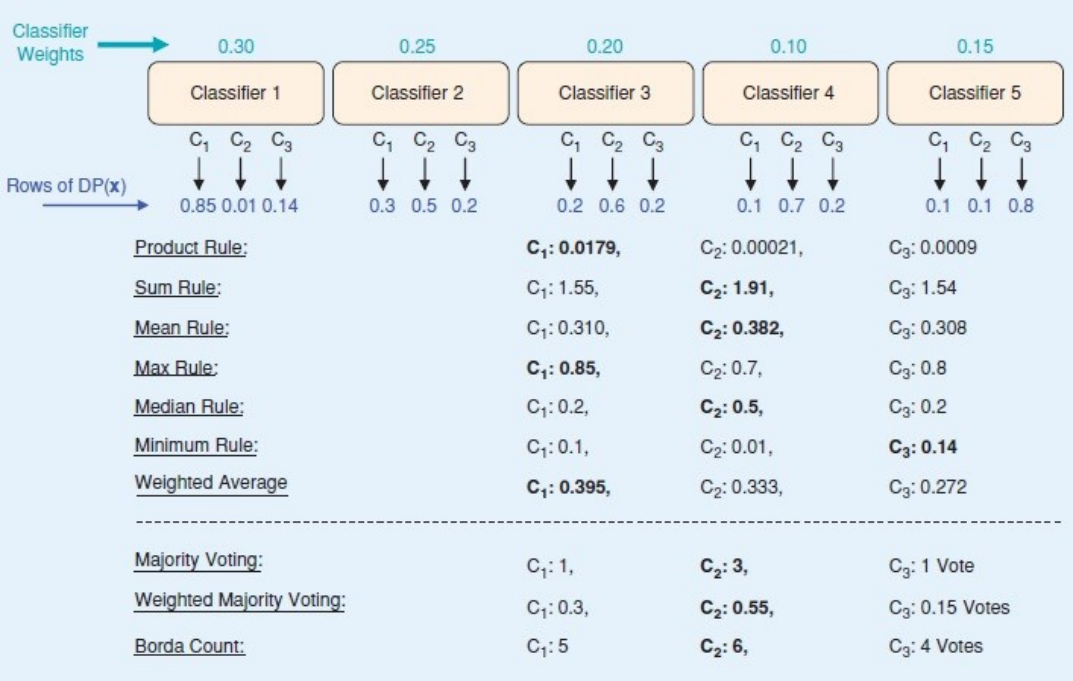

Figura 2.8: Exemplo de várias regras de combinação [Pol06]

Importa salientar a importância desta fase dos classificadores múltiplos: um dos objectivos dos classificadores múltiplos é obter um melhor desempenho relativamente aos classificadores individuais e é exactamente nesta fase que se obtém este resultado, pelo que não faria sentido não ter esta fase.

Deve-se ter em conta que a estratégia de combinação deve maximizar a quantidade de decisões correctas para um dado problema e anular as incorrectas.

No caso de problemas de regressão, a integração é feita utilizando combinação linear, declarado da seguinte forma:

$$\hat{f}_f(x) = \Sigma_{i=1}^{k} [hi(x) * \hat{f}\, i(x)] \tag{2.1}$$

onde hi(x) representam as funções de ponderação.

No contexto deste trabalho, a integração dos modelos, seguiu a classificação de Merz que faz a distinção em dois tipos: funções constantes e não constantes [Mor08]. Para o primeiro caso *hi(x)* é constante, enquanto que para o segundo caso varia conforme os valores de entrada.

Funções de ponderação constantes

Estas funções usam sempre o mesmo conjunto de coeficientes, independentemente dos valores de entrada. Isso constitui uma das grandes desvantagens deste tipo de funções e a razão

pela qual algumas vezes estas funções não são utilizadas, porque ter o mesmo peso em todo o espaço pode não ser adequado em certas situações.

Algumas das funções constantes seguidas por Merz [Mor08] são:

BEM (Basic Ensemble Method), que utiliza como estimador a seguinte igualdade:

$$\hat{f}_{BEM}(x) = \Sigma_{i=1}^{k} [\frac{1}{k} * \hat{f}\, i(x)] \qquad (2.2)$$

Uma outra função seguida por Merz[Mor08] foi proposto por Perrone & Cooper, o GEM (Generalized Ensemble Method), com a seguinte igualdade para o estimador:

$$\hat{f}_{GEM}(x) = \Sigma_{i=1}^{k} [\alpha_i * \hat{f}\, i(x)] \qquad (2.3)$$

Este método apresenta, porém uma desvantagem: a existência de multicolinearidade, que pode ser evitado pela selecção do conjunto. Existe ainda uma abordagem semelhante ao GEM, proposto por Hashem & Shmeiser, que, apesar de sofrer do problema da multicolinearidade, ignora certas restrições. O modelo LR (regressão linear) é também um possível método, apresenta a mesma previsão que o GEM, embora ignora a restrição $\sum^{k}_{j=1}\alpha_i = 1$.

O método *stacked regression*, que foi apresentado por Breiman é baseado no *stacked generalization* de Wolpert. Para este método, dado um determinado conjunto de aprendizagem com M exemplos, o objectivo é obter α_i coeficientes que minimizem

$$\Sigma_{j=1}^{M} [f(x_j) - \Sigma_{i=1}^{k} \alpha_i * \hat{f}\, i(x)]^2 \qquad (2.4)$$

Desta forma, os conjuntos de aprendizagem usados para obter os α_i coeficientes podem ser os mesmos usados para treinar os $\hat{f}_i$, sobrecarregando os dados. Este problema pode ser resolvido através da chamada *v-fold cross validation.*

$$\Sigma_{j=1}^{M} [f(x_j) - \Sigma_{i=1}^{k} \alpha_i * \hat{f}\, i^{(-v)}(x)]^2 \qquad (2.5)$$

Funções de ponderação não-constantes

Estas funções usam diferentes valores para os coeficientes, dependendo da entrada para a previsão, podendo ser estáticas ou dinâmicas. A abordagem estática é definida em termos de tempo de aprendizagem e por sua vez utiliza abordagens diferentes: atribuir modelos a regiões pré-definidas e definir áreas de especialização para cada modelo (selecção estática). A abordagem dinâmica por sua vez é definida em termos de tempo de previsão, onde os pesos são

obtidos em tempo real com base no desempenho dos indicadores da base de dados, a partir do conjunto de treino.

Abordagem por área de especialização

Esta abordagem usa árvores de decisão, que em vez de fazer a previsão, recomendam um indicador de previsão. Foi desenvolvida para a classificação, mas pode ser usada para o problema de regressão desde que se façam algumas alterações na escolha dos meta atributos a serem usados.

Abordagem dinâmica

Nesta abordagem o indicador de selecção é feito em tempo real, isto é, dado um vector de entrada, é escolhido o melhor indicador esperado para realizar determinada tarefa. Ao contrário da abordagem por áreas de especialização, aqui as áreas são definidas em tempo real. A abordagem dinâmica consiste em:

- Dado um valor de entrada, encontrar dados semelhantes do grupo de validação;
- Seleccionar o modelo do conjunto de acordo com o seu desempenho para os dados semelhantes seleccionados;
- Obter a previsão $\hat{f}_i(x)$ para o valor de entrada dado, para cada modelo i seleccionado;
- Obter o conjunto de previsão $\hat{f}_f$. É simples se apenas um modelo for seleccionado, caso contrário é necessário combinar resultados.

O 1º passo, isto é, a obtenção de dados similares é o k-vizinhos mais próximos com a distância euclidiana. Apresenta uma limitação que é a de atribuir o mesmo peso a todas as variáveis independentemente de apresentarem ou não a mesma relevância.

O passo 2 é utilizado de forma a escolher o método com o melhor desempenho de acordo com determinadas métricas.

No passo 4 se mais de um modelo for seleccionado, os resultados são combinados.

Em resumo, as funções de ponderação constantes apresentam a desvantagem de poder ser menos adequada a certas partes do espaço de entrada pelo facto de apresentar o mesmo peso para todo o espaço. A integração de conjunto pode também ser classificada como selecção em que a previsão final é obtida pelo uso de um indicador de previsão ou combinação em que a previsão final é obtida pela combinação de indicadores de dois ou mais modelos.

2.2 Resumo

Classificadores múltiplos

Os classificadores múltiplos constituem um processo complexo, sendo dividido em algumas etapas: geração de modelos, selecção dos melhores modelos e combinação dos modelos para obtenção do resultado final. No entanto, como a selecção de modelos é opcional, alguns autores apenas consideram a geração de modelos e a combinação final. Apesar disso, a utilização da selecção de modelos apresenta algumas vantagens, como por exemplo evita a combinação de modelos muito semelhantes. A geração de modelos tem por finalidade produzir um conjunto de modelos a partir da aplicação de um ou mais algoritmos de indução, sendo que, se for utilizado apenas um algoritmo os modelos dizem-se homogéneos, caso contrário são chamados heterogéneos. A selecção de modelos consiste na selecção de um subconjunto de modelos a partir de um conjunto maior, diminuindo desta forma a complexidade computacional. A combinação final, combina as previsões dos modelos de uma forma estratégica de modo a obter uma melhor previsão do conjunto. Para isso existem as chamadas funções de ponderação utilizadas para esta finalidade. Essas funções podem ser constantes caso utilizem o mesmo peso para as entradas ou não constantes caso utilizem coeficientes diferentes para diferentes entradas.

Uma das características indispensável nos classificadores múltiplos é a diversidade, porque só com diversidade é que se consegue melhores resultados, isto é, se todos os classificadores produzissem a mesma saída era impossível corrigir um determinado erro. Assim, é necessário que os classificadores individuais produzam erros diferentes em diferentes instâncias. Com isto, e após combinar os resultados obtidos com os diferentes classificadores, é possível reduzir o erro total. Nos modelos heterogéneos a diversidade é mais trivial enquanto que nos modelos homogéneos é necessário utilizar algumas técnicas para adquirir diversidade e assim produzir melhores resultados.

O processo de classificadores múltiplos é complexo, mas mesmo assim são vários os métodos destinados a trabalhar sobre eles, embora não se possa dizer que um determinado método é melhor do que outro, pois não existe melhor método para todos os problemas. Alguns dos métodos só se aplicam a problemas de classificação ou só de regressão. São exemplos de classificadores múltiplos *Boosting* e *AdaBoost* utilizados em conjuntos homogéneos e o *Stacked Generalization* utilizados para conjuntos heterogéneos. Geralmente *Boosting* tem atingido um menor erro de teste, e os algoritmos de *Boosting* têm sido considerados mais precisos classificadores sobre uma grande variedade de conjuntos de dados. Uma das grandes vantagens da utilização de classificadores múltiplos é que apresenta uma maior precisão quando comparado com um classificador individual.

3 Ferramentas/Tecnologias

Neste capítulo serão apresentadas as ferramentas que trabalham com classificadores múltiplos bem as que foram utilizadas na realização deste trabalho.

3.1 Ferramentas que trabalham com classificadores múltiplos

Existem algumas ferramentas que implementam os classificadores múltiplos de forma a facilitar a sua utilização. Alguns exemplos destas ferramentas são: SAS Enterprise Miner, SPSS Modeler, RapidMiner, WEKA.

O SAS Enterprise Miner é uma ferramenta utilizada no processo de *data mining* para gerar modelos com alta precisão, com base na análise de grandes quantidades de dados[SAS10]. Implementa *bagging*, *boosting* e conjuntos heterogéneos. Além de trabalhar com grandes quantidades de dados, produz relatórios concisos, incluindo gráficos de fácil interpretação. Esta ferramenta mapeia eficientemente o processo de *data mining* de forma a obter os melhores resultados possíveis no que se refere a construção de modelos.

O SPSS Modeler é uma ferramenta poderosa e versátil que ajuda a desenvolver modelos de previsão exacta de forma rápida e intuitiva, sem programação [SPSS10]. Através de uma variedade de algoritmos pré-construídos, permite a criação de modelos sofisticados de forma fácil e intuitiva.

O RapidMiner considerado a mais completa solução *open-source* de *data mining*[RM10], é uma ferramenta poderosa que oferece uma interface gráfica para criar um pipeline de análise que define os processos analíticos que o utilizador quer que sejam aplicados aos dados. Esta ferramenta além de trabalhar com algoritmos como *boosting* e *bagging*, também trabalha com *stacking*, *random forest* e *rotation forest*, à semelhança do WEKA.

Weka é uma colecção de algoritmos de aprendizagem máquina para tarefas de *data mining* [WEKA10], desenvolvida pela *University of Waikato* em Nova Zelândia. Este software pode ser utilizado tanto em problemas de regressão como de classificação e permite que sejam desenvolvidos modelos de aprendizagem máquina.

3.2 PHP

O PHP é uma linguagem de scripting amplamente utilizada e de propósito geral. É especialmente adequada para o desenvolvimento Web e pode ser incorporada em HTML [PHP10]. É muito utilizada para gerar conteúdos dinâmicos na World Wide Web, como por exemplo, a Wikipédia. Ela foi desenvolvida na década de 90 por Rasmus Lerdorf, influenciada pelas linguagens Perl e C. Tem sofrido muitas actualizações desde então, sendo que, actualmente já suporta orientação a objectos e apontadores. É uma linguagem ideal para instalação e utilização em servidores web, pelo facto de ser extremamente modularizada. O objectivo do PHP é permitir que desenvolvedores web escrevam páginas geradas dinamicamente de forma rápida [PHP10]. Por exemplo, para um site com muita informação e que esta informação precisa ser actualizada sempre, suponhamos a cada hora, o PHP permite fazer esta actualização automaticamente, sem ser necessário fazer tal tarefa manualmente. Neste projecto o PHP foi utilizado para desenvolver o protótipo em geral assim como fazer as devidas ligações de forma a que fosse possível executar o código escrito na linguagem R.

3.2.1 Características

Algumas das principais características do PHP são:

- a portabilidade – isto é, é independente de plataforma. O PHP funciona em vários sistemas operativos. Alguns exemplos são: Windows, Linux, MAC OS, Solaris.
- velocidade e robustez – o PHP mostra-se uma linguagem rápida no que se refere ao processamento da informação e ainda é robusto.
- estruturação e orientação a objectos – nos dias de hoje o paradigma de programação orientada à objectos está predominante nas linguagens de programação e aos poucos tem sido utilizada em vez da programação procedimental. Neste tipo de programação os objectos têm propriedades e funções que são executados.
- software gratuito de código aberto – o PHP é um software gratuito pelo que pode ser encontrado em www.php.net para instalação. Isso faz com que seja mais fácil a sua aquisição e as pessoas optem mais por utilizá-lo. Além disso é de código aberto, o que faz com que muitas pessoas o utilizem pois a resolução de problemas que advém da sua utilização torna mais fácil e prática e ainda garante uma certa flexibilidade.
- Baseado no servidor – o código escrito em PHP é executado no servidor e o seu resultado é mostrado no browser. Assim sendo o cliente fica mais leve e as páginas

serão executadas de imediato o que leva a um desenvolvimento de sistemas mais complexos, mas ao mesmo tempo de fácil e rápido acesso aos utilizadores.

- Acesso à base de dados – o PHP pode aceder virtualmente qualquer base de dados[PHPDB10]. Sendo assim, suporta diversas bases de dados, como por exemplo: MySQL, PostgreSQL, Oracle, SQL Server, em que para cada uma delas existe uma variedade de funções que podem ser aproveitadas no desenvolvimento de código.
- Interacção com HTML – o PHP pode gerar HTML e o HTML pode passar informações ao PHP [PHTML10]. O PHP e o HTML podem ser facilmente juntados, sendo assim, ao escrever uma página em PHP é normal e comum encontrar partes de código escritas em HTML. Um exemplo pode ser visto na figura a seguir:

```
<html>
<head>
    <title>Teste de algoritmos de previs&atilde;o</title>
</head>
<body>

<?php
    echo "Protótipo de teste.";
?>
</body>
</html>
```

Figura 3.1: Exemplo de código PHP embutido em HTML.

3.2.2 Vantagens e Desvantagens

O PHP tal como as outras linguagens de programação apresenta vantagens e desvantagens. Como principais vantagens destacam-se:

- velocidade de processamento;
- software livre;
- eficiência;
- código fonte livre e licença gratuita;
- orientação à objectos;
- possibilidade de executar em qualquer plataforma;
- muita documentação disponível entre os quais manuais, existência de wiki;
- flexibilidade;

- facilidade de aprendizagem;
- simplicidade de uso;
- facilmente extensível;
- apoio de grande número de criadores.

As suas principais desvantagens são:

- O facto de ser uma linguagem para desenvolvimento web faz com que esteja em desvantagem relativamente as linguagens de propósito geral;
- O facto de ser de hospedagem gratuita faz com com possa haver desvantagem relativamente à confiança e imagem do site por parte dos utilizadores;
- O PHP permite código embutido em HTML e isso leva a que haja pedaços de código PHP, seguido de trecho HTML, uma ou várias vezes. Esta mistura de códigos pode fazer com que o código como um todo esteja confuso, levando mesmo os utilizadores a terem problemas com a interpretação;
- O facto de ser de hospedagem gratuita faz com que haja limites máximo estabelecidos para o tamanho de um arquivo.

3.3 Linguagem R

R é uma linguagem e um ambiente de desenvolvimento integrado para cálculos estatísticos e gráficos [R10]. Surgiu na década de 90 por Ross Ihaka e Robert Gentleman e chama-se “R” exactamente por causa dos nomes dos seus criadores. O seu código fonte está sob a licença GNU GPL. O R fornece uma ampla variedade de estatísticas (modelação linear e não linear, testes estatísticos clássicos, análise de séries temporais, classificação, análise de aglomerados) e outras técnicas gráficas, sendo altamente extensível [R10]. Para muitos estatísticos, o R é considerado uma linguagem particularmente útil pelo facto de possuir um conjunto de mecanismos para a organização de dados, execução de cálculos sobre a informação e criar representações gráficas de conjuntos de dados [NYT09]. O seu principal objectivo é implementar algoritmos para processar dados. O R apresenta algumas semelhanças com outras linguagens de programação, por exemplo: C, Java e Perl no que se refere a execução de grande variedade de tarefas de computação. No trabalho em questão o R é utilizado para desenvolver os algoritmos a serem testados, assim como o código que faz a geração e selecção dos modelos bem como a produção do resultado final, ou seja, o código correspondente ao desenvolvimento de classificadores múltiplos. Todo este código lida com grandes quantidades de dados, entre os quais dados analíticos, o que o torna ideal para o desenvolvimento do trabalho em questão.

3.3.1 Características da linguagem R

São apresentadas algumas das principais características da linguagem R:

- O sistema básico do R é complementado por vários “pacotes recomendados” que fazem parte da distribuição padrão do R [FA05];
- O R é um software de alta qualidade no que diz respeito aos softwares de cálculos estatísticos. No seu desenvolvimento estão associadas algumas das principais figuras da computação estatística, como por exemplo: Douglas Bates, John Chambres, Brian Ripley e Luke Tierney [FA05];
- Apresenta uma interface gráfica que permite o desenho de gráficos de alta qualidade;

3.3.2 Vantagens e desvantagens da linguagem R

São apresentadas de seguida as principais vantagens da utilização da linguagem R:

- pouco gasto de memória para execução;
- velocidade de processamento;
- código fonte aberto;
- o R corre em vários sistemas operativos como por exemplo: Windows, Macintosh e Unix/Linux.
- O R possui um sistema de ajuda que permite obter informação extra sobre a linguagem, bem como aspectos que lhe são associados [Tor06]. Esta ajuda pode ser conseguida através da utilização da expressão “*Help*” na janela de execução do R. Além disso possui uma vasta documentação, bem como *Faq* e *mailing list*.
- É um software de fácil utilização.

Algumas das principais desvantagens da utilização da linguagem R são apresentadas a seguir:

- Carência de interfaces gráficas simples e eficientes, o que faz com que seja considerado um ambiente computacional difícil de aprender e usar.
- Os objectos do R são guardados na memória do computador, pelo que existe um limite de espaço na memória.

3.4 MySQL

O MySQL é um sistema de gestão de base de dados que utiliza o SQL(Structured Query Language) como interface. Foi criado na Suécia por três autores, sendo dois suecos e um finlandês. Alguns dos seus utilizadores são: HP, Nokia, Sony, Google, etc. O MySQL é

facilmente integrável no PHP, sendo que hoje em dia, é automaticamente oferecido nos pacotes de hospedagem de sites na Internet. O MySQL tornou-se a base de dados mundial de código aberto mais popular do seu alto desempenho, alta confiabilidade e facilidade de uso [WMySQL11]. O MySQL corre em várias plataformas, tais como: Linux, Windows, Mac, Solaris, HP-UX, IBM AIX [WmySQL11], permitindo assim uma grande flexibilidade ao utilizador.

O MySQL disponibiliza aulas e cursos como por exemplo:

- MySQL para iniciantes;
- MySQL para administradores de bases de dados;
- MySQL para desenvolvedores;
- MySQL performance *tuning*;
- MySQL alta disponibilidade;
- MySQL *cluster*;
- Desenvolvedor Técnicas MySQL;
- MySQL e PHP – Desenvolvendo aplicações web dinâmicas;
- MySQL avançado *Stored Procedures*.

3.4.1 Características

Algumas das características do MySQL e que são consideradas úteis no contexto desta dissertação são:

- Portabilidade – funciona em várias plataformas como: Windows, Linux, MAC OS, Solaris. Escrito em C e C++. Tem APIs disponíveis para: C, C++, Eiffel, Java, Perl, PHP, Python, Ruby e Tcl [DocMySQL11];
- Tem um completo suporte a operadores e funções nas partes SELECT e WHERE das consultas [DocMySQL11];
- È possível misturar tabelas e bases de dados diferentes nas mesmas consultas [DocMySQL11].
- Nomes das funções não conflituam com nomes de tabelas ou colunas [DocMySQL11];
- Segurança – possui um sistemas de privilégios e *passwords* seguro que permite verificação baseada em estações/máquinas e as *passwords* são criptografadas quando o utilizador conecta ao servidor [DocMySQL11];
- Escalabilidade e limites – lida com bases de dados com grandes quantidades de dados e são permitidos até 32 índices por tabela;
- Pouco exigente quanto a recursos de hardware

- facilidade de uso
- é um software livre com base na GPL(General Public License)
- Todos os programas MySQL podem ser chamados com as opções *–help* ou -? para obter ajuda online;
- Suporta controle transaccional, gatilhos, cursores, procedimentos armazenados e funções;
- Replicação facilmente configurável;
- Interface gráficas (MySQL Toolkit) de fácil utilização cedidos pela MySQL Inc.

3.4.2 Vantagens e desvantagens

Algumas das principais vantagens do MySQL são:

- É simples, ou seja, é fácil de aprender e usar, relativamente à outras bases de dados;
- É um software *open-source*;
- É bem suportado, tendo serviços de suporte e manutenção como por exemplo actualização de código e correcção de bugs;
- Flexível e escalável, permitindo mais opções para recursos adicionais, tais como mecanismos de armazenamento;
- É considerado a base de dados mais rápida;
- É seguro, permitindo controlo de utilizadores e acesso, e, se necessário verificação e correcção de tabelas corrompidas;
- Tanto pode ser usado para aplicações na Internet como para aplicações desktop;
- É gratuito.

As suas principais desvantagens são:

- Nem sempre o MySQL tem total apoio aos procedimentos armazenados;
- No que se refere a recuperação, pode haver problemas se o sistema desligar inesperadamente, podendo o documento perder-se;
- utiliza a GPL. Apesar de ser bom para muitos, em certos ambientes não é o melhor, sendo preferível licenças comerciais.

3.5 Resumo e Conclusões

São notáveis as vantagens do PHP em conjunto com a base de dados MySQL para desenvolvimento de aplicações web assim como as vantagens da linguagem R para cálculos estatísticos, apesar das desvantagens que apresentam.

O PHP tem sido cada vez mais utilizado no desenvolvimento de aplicações web dada a sua facilidade de aprendizagem e as suas funcionalidades que permitem executar inúmeras tarefas de forma eficiente.

O R, dado as suas potencialidades no que se refere aos cálculos estatísticos, algoritmos com manipulação de dados, tem-se tornado numa ferramenta muito utilizada, vindo a ganhar destaque por utilizadores comuns, mas também no mundo empresarial.

O MySQL tem sido muito utilizado nos últimos tempos, tendo-se tornado uma escolha frequente de muitas empresas e utilizadores comuns. Funciona muito bem com o PHP, sendo que o último possui mesmo funções destinadas a fazer a interligação entre estas duas linguagens.

Desta forma, a melhor solução para o desenvolvimento do projecto em causa foi a utilização do PHP em conjunto com a linguagem R, utilizando a base de dados MySQL. No que se refere a interligação entre estas duas linguagens para o desenvolvimento do presente trabalho não houve grandes dificuldades, porque a partir do PHP é possível executar código escrito em outras linguagens e o R faz parte deste conjunto de linguagens. Sendo assim, a implementação do trabalho foi feita em PHP, apresentando uma interface web para os utilizadores, e por detrás disso está o R onde é processado o código relativo ao desenvolvimento dos classificadores múltiplos. Em relação ao MySQL só foi preciso fazer a ligação/conexão com o PHP, através de instruções existentes no PHP, pelo que também não houve problemas.

A combinação das linguagens PHP e R no desenvolvimento do trabalho mostra uma outra vantagem: apresenta uma interface com o utilizador fácil de utilizar e de perceber, o que faz com que o utilizador só tem de se preocupar em fornecer os dados pedidos. Tendo em conta que quem vai utilizar esta aplicação são pessoas que percebem sobre o assunto “classificadores múltiplos”, estes não terão problemas em fornecer os dados. A aplicação pode apresentar problemas ao utilizador, caso este forneça os dados incorrectamente. Posto isso, só terá de esperar para ver o resultado. Isso pode levar muito tempo, podendo mesmo levar dias, porque a aplicação está mesmo desenvolvida para grandes quantidades de dados, embora possa demorar pouco tempo, caso a quantidade de dados não seja muito grande.

4 Implementação

Neste capítulo são apresentados os detalhes referentes à implementação, assim como o funcionamento da aplicação.

4.1 Funcionamento da aplicação

O funcionamento da aplicação desenvolvida segue o esquema representado no diagrama de actividades que se segue.

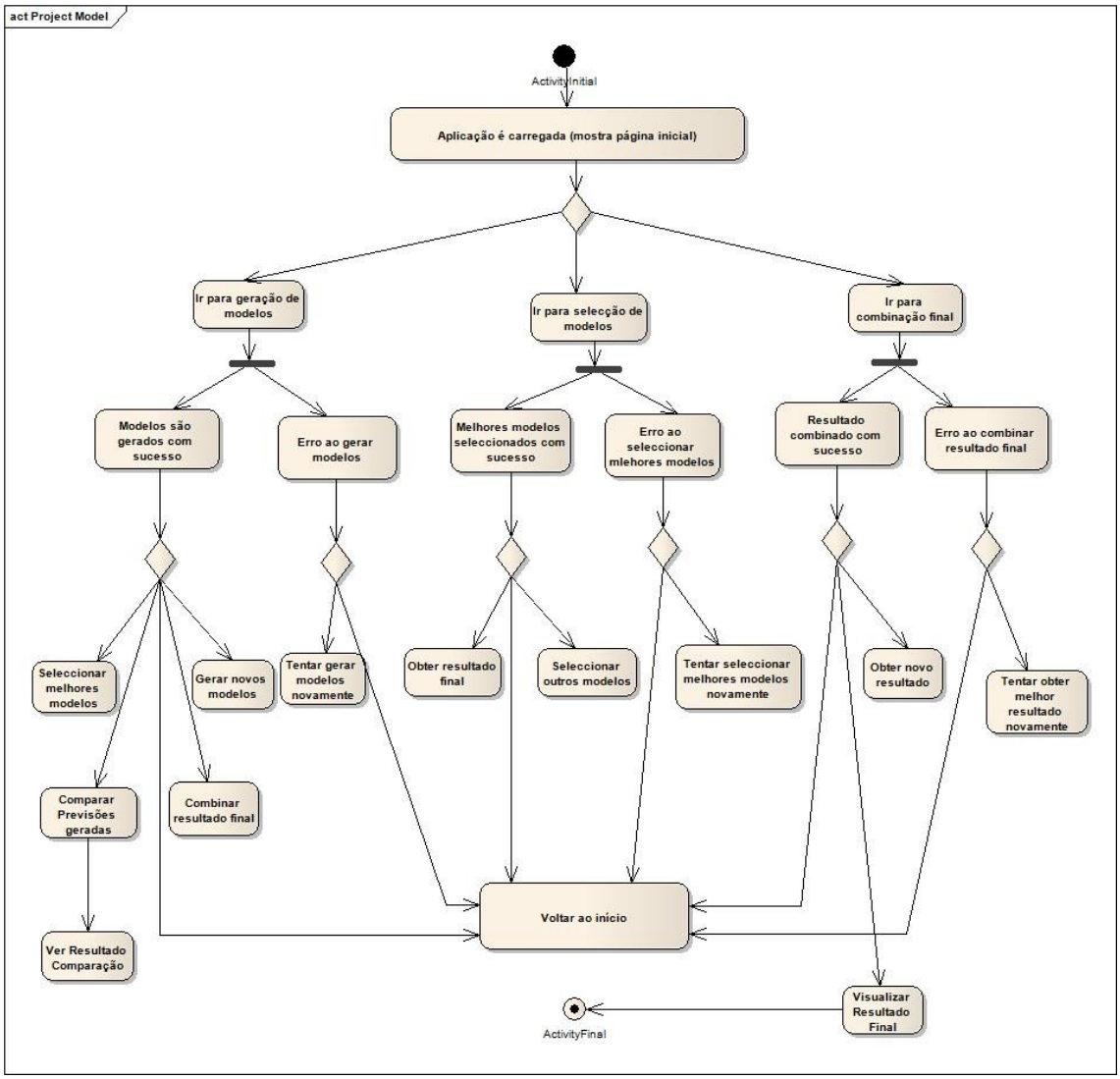

Figura 4.1: Diagrama de actividades do funcionamento da aplicação

A execução de cada uma das três fases tem como resultado o sucesso quando corre tudo dentro da normalidade ou insucesso quando algum dado necessário ao bom funcionamento não está de acordo com o especificado. Qualquer que seja o estado em que a aplicação estiver, é sempre possível voltar ao início, isto é, quer esteja num estado de erro, quer esteja num estado de sucesso, pode-se sempre voltar ao início. Cada uma das fases produz um resultado que será guardado. Assim sendo, caso um utilizador pretenda executar a geração de modelos num determinado dia, e só executar as outras fases depois, pode fazê-lo sem problemas porque o resultado da primeira fase está guardada e poderá ser acedida a qualquer momento. Mas o processo só termina quando todas as três fases forem executadas e todas retornarem sucesso para o caso do utilizador pretender usar a aplicação com todas as três fases que fazem parte dos classificadores múltiplos ou quando executar a primeira e a terceira fase e ambas retornarem sucesso para o caso do utilizador pretender usar simplesmente a geração de modelos e a combinação final. Isso porque não faz sentido por exemplo fazer a combinação final sem ter feito a geração de modelos, pelo que o resultado de uma fase anterior é aproveitado na fase

seguinte. Para além de executar as três fases dos classificadores múltiplos, a aplicação permite a comparação de previsões do modelos gerados. Sendo assim, ao terminar a primeira fase com sucesso, caso o utilizador queira, ele pode escolher a opção “Comparar Previsões” e ver o resultado da comparação. Esta comparação é feita tendo em conta o valor real e o valor da previsão gerada pelo modelo.

Para que a aplicação funcione é preciso que o computador onde a aplicação é executada contenha a linguagem R instalada.

4.2 Organização do código

A organização do código em PHP foi dividida em várias secções de forma a ter uma melhor estruturação. Desta forma ficou dividida em:

- uma pasta css contendo o ficheiro css responsável pelo estilo de como as páginas ficariam;
- uma pasta database contendo as classes correspondentes às classes existentes na base de dados;
- uma pasta chamada includes que contém o ficheiro startup.php responsável por fazer a inicialização da aplicação, assim como fazer as devidas conexões entre o PHP e MySQL;
- uma pasta template contendo os ficheiros com código html/xml;
- uma pasta upload onde são guardados os ficheiros feitos upload;
- finalmente na raíz ficaram guardados dois tipos de ficheiros .php: os primeiros correspondem aos nomes pelos quais serão chamados no browser na altura da execução. Estes ao serem executados chamam os ficheiros guardados na pasta template que por sua vez vão chamar os segundos ficheiros guardados na raiz, isto é, os com o código php responsável por executar as acções, os chamados ficheiros com a acção. Para melhor esclarecer este ponto, segue-se um exemplo: ao gerar os modelos, foi criado um ficheiro geraModelos.php que é aquele que é chamado quando se vai executar a aplicação. Este ficheiro chama o geraModelos.tpl que está guardada na pasta templates e por usa vez vai chamar o accao_geraModelos.php que contém todo o código referente a geração de modelos.

Desta forma, há uma melhor organização porque o código não fica como um todo, evitando que fique confuso e caso surja algum erro ou problema é mais fácil identificar as razões e resolvê-lo.

4.3 Solução implementada

A solução implementada segue a estrutura natural do processo de classificadores múltiplos, isto é, a existência de três fases sequenciais: geração, selecção e combinação de resultados, sendo que nas duas últimas fases é aplicada a validação cruzada (*cross validation*). Esta decisão

deve-se ao facto de ser melhor particionar o problema complexo em pequenos sub-problemas. E ainda, pelo facto de poder ir testando cada uma das fases logo a seguir à sua implementação. Desta forma ao implementar a primeira fase, isto é, a geração de modelos, foi possível testá-la logo e verificar se a implementação foi bem sucedida, sem ter que esperar pela implementação total do projecto, uma vez que ao surgir qualquer erro, seria mais difícil detectá-lo e corrigi-lo.

A implementação foi o desenvolvimento de uma aplicação web que foi dividida em três partes: (1) uma primeira em que após serem fornecidos os dados necessários para a geração dos modelos, estes eram gerados; (2) seguidamente são seleccionados os melhores modelos a partir dos modelos gerados na fase anterior; (3) finalmente é feita a combinação utilizando uma função de combinação de forma a produzir o resultado final a partir dos melhores modelos seleccionados anteriormente. Cada uma das três fases de desenvolvimento do trabalho foi dividida em outras duas partes: primeiro a construção de uma base de dados desenvolvida no MySQL para armazenar as informações relevantes para a respectiva etapa e em segundo lugar a implementação da interface.

4.3.1 Geração de modelos

A fase de geração é a primeira fase de implementação do projecto e corresponde à geração de modelos que em cada processo de classificadores múltiplos dá origem ao resultado final. Cada modelo é gerado pela aplicação de um determinado algoritmo, com determinados parâmetros e um certo conjunto de dados. Estes dados estão contidos num ficheiro, o *DataSet*. Portanto, para a geração do conjunto de modelos é necessário passar os seguintes parâmetros: os algoritmos que pretendemos testar, a designação dos parâmetros, uma lista de listas com a gama de variação dos parâmetros e um *DataSet*. Por exemplo, um dos algoritmos de teste utilizados foi o *svm linear* que tem como parâmetros *cost* e *nu*, e, portanto para o testar com os valores 0.05, 0.06 e 0.1, 0.2. 0.4, respectivamente, a lista deverá ser formada por duas listas, sendo a primeira (0.05, 0.06) e a segunda (0.1, 0.2, 0.4). Os algoritmos de teste são escritos na linguagem R e o resultado produzido terá o formato RData.

Processamento

O processamento é feito da seguinte forma: existe uma função/algoritmo principal responsável por gerar os modelos. Esta função recebe os seguintes parâmetros:

- os algoritmos de teste separados por vírgula – como exemplo de algoritmos de teste temos: o *svm*(support vector machine) e o *ppr*(projection pursuit regression);

- os parâmetros dos algoritmos de teste fornecidos em listas de listas. Os dados contidos nestas listas referentes à cada algoritmo de teste são combinados de forma a que o respectivo algoritmo seja executado para todas as combinações possíveis;
- um *DataSet* que contém os dados de teste;
- as variáveis do *DataSet* assim como o valor que devem tomar. Este valor pode ser *TRUE* ou *FALSE*;
- uma função que será responsável por gerar o setup experimental. Como exemplos temos TenFoldCrossValidation que utilizando a validação cruzada faz a divisão em dez partes;
- os parâmetros da função responsável por gerar o setup experimental. Estes parâmetros variam de acordo com a função utilizada;
- duas *flags* que devem tomar o valor *TRUE* ou *FALSE*. Estão relacionadas com o facto do utilizador querer executar a partição do conjunto inicial.

O código da função responsável por gerar o setup experimental assim como todo o código referente à geração de modelos são escritos na linguagem R.

Numa interface com o utilizador desenvolvida em PHP, é pedido que sejam fornecidos os argumentos da função principal assim como os seguintes dados:

- o caminho do executável da linguagem R (R porque é nesta linguagem que o código a ser processado é escrito);
- o directório para onde serão feitos os uploads. Neste directório serão guardados os modelos gerados, assim como outro resultado que tenha sido obtido em alguma fase. E, caso, o utilizador queira fazer upload dos ficheiros contendo os códigos escritos em R, estes serão armazenados neste directório.
- A variável na qual o utilizador pretende guardar o resultado.

Fornecidas estas informações, o PHP faz o seu processamento. É executada uma função que retorna um array contendo os algoritmos de teste, em que cada elemento do array corresponde a um algoritmo. Para cada um desses algoritmos é chamada uma função que faz a combinação dos seus parâmetros de forma a determinar o número de vezes que a função principal deve executar o respectivo algoritmo de teste. Esta função retorna um array em que cada elemento corresponde a uma execução do algoritmo. Para exemplificar consideremos o algoritmo de teste *svm* linear que tem dois parâmetros *cost* e *nu* com os valores 0.05 e 0.06 para *cost* e 0.1, 0.2 e 0.4 para *nu*. A função responsável por fazer a combinação retorna um array com seis elementos e a função principal deverá executar o *svm* para todos eles. Os elementos que fazem parte do array resultante são:

- o primeiro elemento: *cost*=0.05, *nu*=0.1;
- o segundo: *cost*=0.05, *nu*=0.2;
- o terceiro: *cost*=0.05, *nu*=0.4;
- o quarto: *cost*=0.06, *nu*=0.1;

- o quinto: *cost*=0.06, *nu*=0.2;
- o sexto elemento: *cost*=0.06, *nu*=0.4.

Depois de executada a função descrita anteriormente é executada a função principal conforme as combinações determinadas. Se tudo correr bem são gerados os modelos que formam o resultado da primeira fase do projecto. O processo termina com a produção dos modelos. Finalmente são guardadas na base de dados as informações sobre os modelos gerados, o conjunto formado pelos modelos, o software usado, a função principal, os algoritmos de teste, os parâmetros e o setupExperimental. O modelo em si, isto é o objecto em formato *RData* é guardado no directório destinado aos uploads. A tabela seguinte mostra alguns dos algoritmos que podem ser usados para teste bem como os seus parâmetros:

Tabela 4.1: Alguns algoritmos de teste e seus parâmetros

Algoritmo	*Parâmetro1*	*Parâmetro2*	*Parâmetro3*	*Parâmetro4*
svm linear	cost	nu	-	-
svm radial	cost	nu	gamma	-
svm sigmoid	cost	nu	gamma	coef0
ppr supsmu	nterm	optlevel	bass	span
ppr spline	nterm	optlevel	df	-
ppr gcvspline	nterm	optlevel	gcvpen	-
lm	-	-	-	-
rpart	-	-	-	-
rf	mtry	-	-	-

Diagrama de classes da geração de modelos

A seguir é apresentado o diagrama de classes correspondente à fase de geração de modelos.

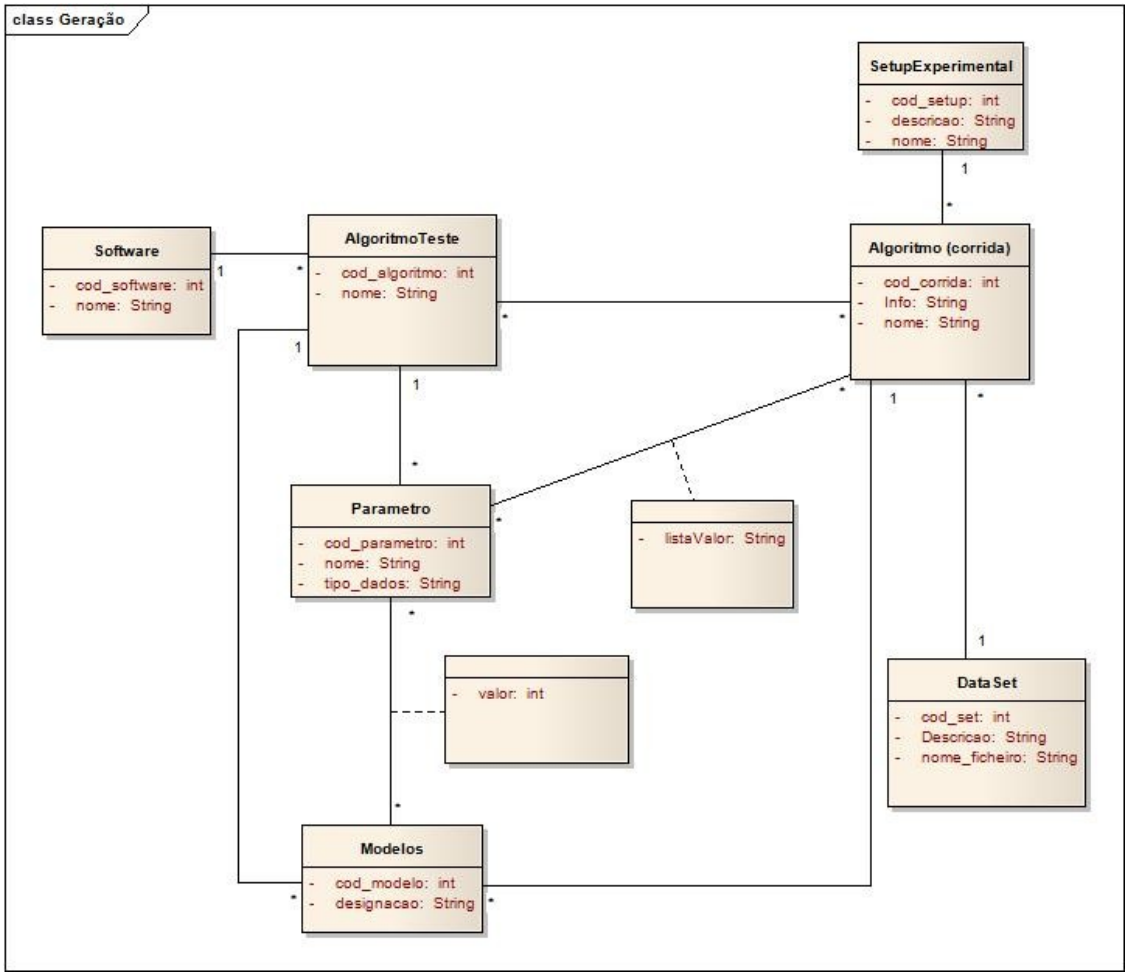

Figura 4.2: Diagrama de classes da geração de modelos

A classe Software corresponde ao software em que é escrito o código a ser executado pelo PHP, neste caso o R. AlgoritmoTeste é qualquer um dos algoritmos de teste a serem utilizados para gerar os modelos e têm Parâmetros que são usados para fazer os testes. Estes algoritmos são utilizados pela Corrida que é uma função principal em que um dos seus argumentos é o *DataSet*. O *DataSet* é um conjunto de dados contendo os dados que serão usados para gerar os modelos. Modelos são os modelos gerados na primeira fase, constituem o resultado desta fase e são usados nas fases posteriores. Existe ainda o SetupExperimental que é gerado por uma função e é usado para gerar os modelos. Como exemplo temos um setup experimental gerado por validação cruzada através da partição do conjunto inicial em *k* partes, isto é, *k-fold* .

Comparação das previsões obtidas

Depois de executada a geração dos modelos e antes de passar para a selecção dos melhores modelos, o utilizador pode escolher fazer a comparação de previsões geradas pelos modelos com o valor real. Esta comparação é feita utilizando uma função escrita em R e pode ser por exemplo o cálculo do erro médio quadrado ou o cálculo do erro médio absoluto.

Para executar tal comparação é preciso fornecer os seguintes dados:

- O directório para upload;
- o ficheiro de entrada que contém a função que executa a comparação;
- o *DataSet* utilizado na fase de geração de modelos;
- as variáveis do respectivo DataSet;
- os argumentos da função de comparação;
- a variável para guardar o resultado final;

Depois de fornecidos os dados, a função é executada e o valor é guardado num ficheiro RData no directório especificado anteriormente.

4.3.2 Selecção de modelos

Uma vez produzidos os modelos passamos para a fase de selecção que, à semelhança da primeira fase, tem como dados de entrada um algoritmo e os seus parâmetros. Neste caso devem ser passados os modelos produzidos anteriormente. Dados estes inputs, é feita a selecção dos melhores modelos. Esta selecção é feita tendo em conta vários critérios que dependem das funções a serem utilizadas que são passadas pelo utilizador. Alguns dos critérios que se costumam utilizar são: diversidade, precisão, etc. Nesta fase o resultado é um conjunto de modelos, embora mais restrito do que o conjunto produzido na fase anterior, pois agora apenas fazem parte deste conjunto os melhores modelos segundo a função usada. O resultado é apresentado sob a forma de números inteiros que representam os índices dos modelos produzidos na fase anterior.

Processamento

Nesta fase o processamento é feito da seguinte forma: é apresentada uma janela de interface com o utilizador, onde são pedidos os seguintes dados:

- algoritmo de selecção - é uma função à escolha do utilizador que deverá ter certos critérios para escolher os melhores modelos;
- argumentos do algoritmo de selecção – estes argumentos dependem de cada algoritmo passado. Sendo assim tanto o número de argumentos, como o nome de cada um e o valor são variáveis;
- o ficheiro onde se encontra o algoritmo de selecção uma vez que o algoritmo está escrito num ficheiro separado;

- uma *combobox* para que seja escolhido o conjunto de modelos que se pretende utilizar – a *combobox* apresenta todos os conjuntos de modelos que já tenham sido gerados e o utilizador escolhe o conjunto que deseja utilizar;
- A variável onde se pretende guardar o resultado.

Ao serem passadas as informações necessárias, o ficheiro contendo a função é executado retornando um resultado que é uma lista de números que correspondem aos modelos seleccionados. Depois disso é chamada uma função desenvolvida em PHP que ao receber o resultado da execução anterior retorna os modelos correspondentes. Estes modelos constituem o *Ensemble*, isto é um conjunto mais restrito formado apenas pelos melhores modelos gerados na fase anterior. Depois disso, o *Ensemble* é guardado na base de dados e será utilizado na fase seguinte. Se na altura de processamento dos dados pelo PHP, correr tudo bem é sinal de que os melhores modelos foram seleccionados com sucesso, de acordo com o algoritmo utilizado. Caso contrário, dá um erro, sendo necessário rever se os dados foram passados correctamente.

Diagrama de classes da fase de selecção de modelos

A seguir é apresentado o diagrama de classes referente a fase de selecção de modelos.

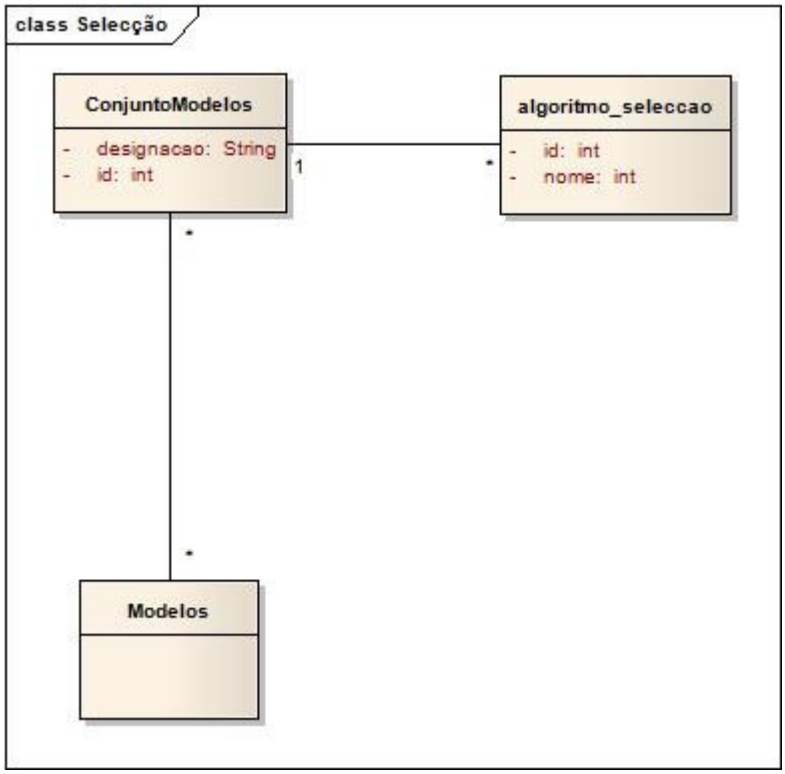

Figura 4.3: Diagrama de classes da fase selecção de modelos

ConjuntoModelos é referente ao conjunto de modelos gerado na fase de geração de modelos que é constituído por vários Modelos. Nesta fase temos um algoritmo_seleccao que é uma função utilizada para seleccionar os melhores modelos de entre todos os que foram produzidos e varia de acordo com os objectivos do utilizador. Os Modelos são todos os gerados anteriormente que fazem parte de um determinado conjunto de modelos.

4.3.3 Combinação

Uma vez seleccionados os melhores modelos, é feita a combinação de onde se obtém o resultado final. Para tal existe um novo algoritmo ao qual são dados os modelos que serão usados para fazer a previsão final. Estes modelos podem ser obtidos de uma de duas formas:

- Obtidos da fase de selecção. Neste caso os modelos são os considerados melhores. Isto acontece quando o processo de classificadores múltiplos passa por todas as três fases.
- Obtidos directamente da fase de geração, isto é, neste caso o processo de classificadores múltiplos passa apenas pelas fases de geração do conjunto de modelos e combinação, sendo que os modelos são todos os produzidos inicialmente.

A combinação dos modelos segue uma abordagem segundo funções de ponderação constante e não constante, descritas anteriormente, conforme for o caso a aplicar. O output desta fase é o output final do processo de classificadores múltiplos, isto é a previsão final, num ficheiro RData. Dependendo do problema ser de regressão ou classificação, este output será uma variável numérica ou não numérica, respectivamente.

Processamento

O processamento da combinação final é feito de duas formas diferentes: combinação dos modelos obtidos directamente da fase de geração ou combinação dos melhores modelos obtidos a partir da fase de selecção.

No primeiro caso, isto é, para a combinação dos modelos obtidos directamente a partir da fase de geração é necessário fornecer:

- nome do ficheiro que contém o código a ser executado em R via PHP. Este ficheiro pode estar guardado tanto no directório para upload como noutro directório;
- a função de combinação – função responsável por fazer a combinação dos resultados produzidos pelos modelos e produzir um único resultado para todo o sistema. Utiliza uma determinada estratégia de forma a que ao fazer a combinação, seja produzido um

resultado final melhor do que os resultados produzidos individualmente por cada modelo;

- argumentos da função de combinação – estes parâmetros variam de função para função.
- Variável para guardar o resultado da execução.

Fornecidos os dados, estes são processados pelo PHP que faz a interacção com o R de forma a que o código seja executado e finalmente é produzido o resultado final.

No segundo caso, isto é, a combinação dos melhores modelos, para além dos dados referidos anteriormente é necessário fornecer também o *ensemble*, isto é, o conjunto formado pelos melhores modelos. Este é escolhido a partir de uma *combobox* que apresenta todos os *ensembles* gerados até o momento. Depois disso, a função de combinação é chamada via PHP. Esta vai utilizar os respectivos argumentos e o *ensemble* para produzir o resultado final.

Diagramas de classes da fase combinação final

Nesta fase são apresentados dois diagramas de classes: um para o caso dos modelos usados serem obtidos directamente da primeira fase e outro para o caso de serem obtidos da segunda fase.

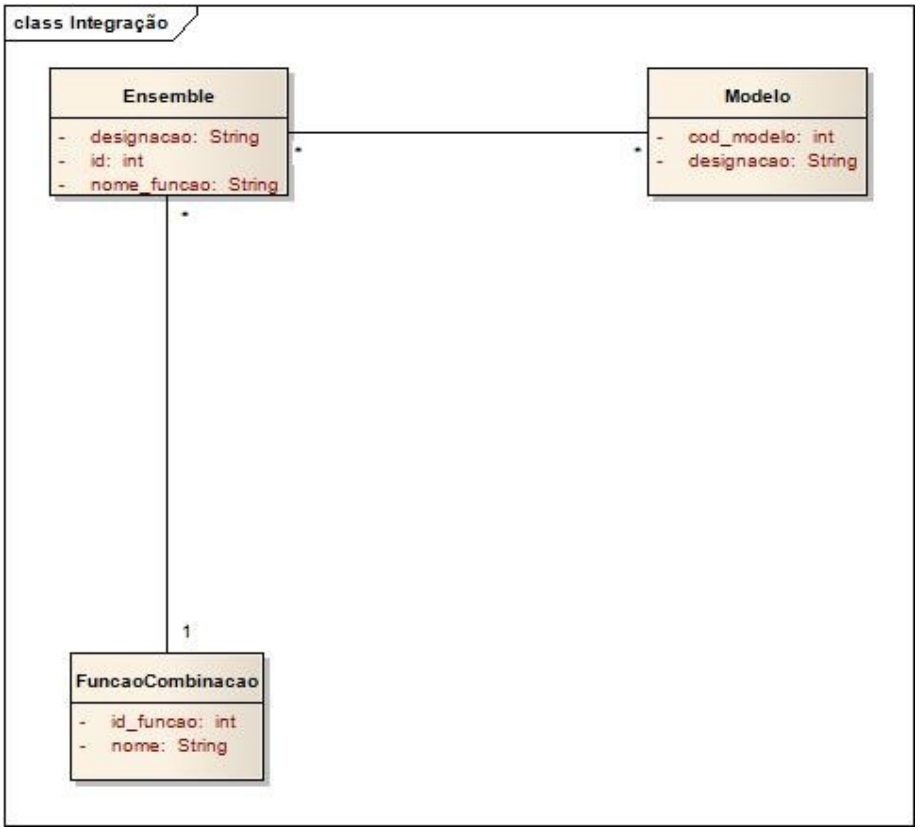

Figura 4.4: Diagrama de classes da fase combinação final sem passar pela fase de selecção

O *Ensemble* é o subconjunto seleccionado na fase anterior contendo os melhores modelos. A FuncaoCombinacao é a função utilizada para fazer a combinação e assim gerar o resultado final.

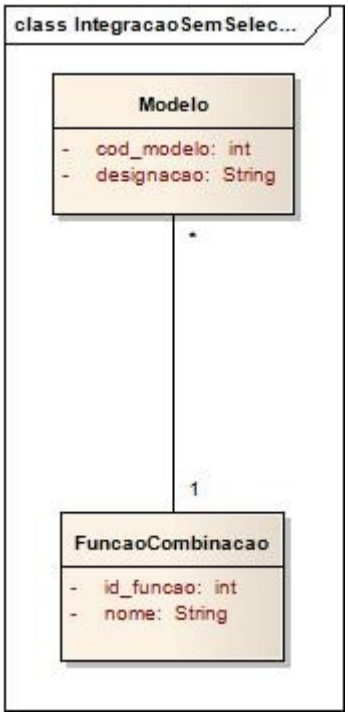

Figura 4.5: Diagrama de classes da fase combinação final passando pela fase de selecção

Neste caso só faz sentido ter as classes Modelo e FuncaoCombinacao em que a função de combinação utiliza os modelos para produzir o resultado final.

Apresentação do resultado final

Depois de executada a terceira fase e de ser produzido um resultado final único para todo o processo, o utilizador pode escolher a opção "Ver resultado", sendo necessário fornecer o nome do ficheiro que contém o resultado e o directório onde este se encontra. Posto isso, são apresentadas as opções de abrir o ficheiro ou fazer download do mesmo e o utilizador escolhe a que pretender. Com isso o processo chega ao fim.

4.4 Conclusões

A implementação da solução foi dividida em três fases devido à complexidade do problema dos classificadores múltiplos. Desta forma, ao implementar cada uma das fases em separado seria mais fácil perceber e ultrapassar as dificuldades que fossem aparecendo no decorrer do trabalho. Durante o desenvolvimento foram utilizadas três ferramentas: o PHP, a linguagem R e a base de dados MySQL. De todos eles o que menos trouxe problemas foi o MySQL, em que as bases de dados foram criadas sem dificuldades e depois só foi necessário fazer as devidas ligações/conexões entre o PHP e MySQL.

A primeira fase foi a que demorou mais tempo e deu mais trabalho pelo facto de ter mais código para desenvolver e ser necessário familiarizar-se com a linguagem R, visto que até ao momento só tinha tido contacto com o PHP. Além de familiarizar-se com a linguagem R, foi preciso estudar mais o PHP no sentido de saber como fazer a interacção entre as duas linguagens. A base de dados referente a esta fase é a maior porque tem mais dados para guardar além de ser nesta fase onde são pedidas mais informações, sendo que algumas delas são utilizadas nas fases seguintes. Posto isso, foi uma questão de explorar mais sobre as duas linguagens e ir desenvolvendo o código. O tempo de execução desta fase é muito variável, dependendo muito do número de modelos gerados e do tamanho do *DataSet*, quer em termos de número de instâncias como em número de variáveis/atributos.

A segunda fase, isto é, a selecção de modelos, demorou menos porque já estava familiarizada com a linguagem R e os conhecimentos do PHP estavam mais aprofundados. Nesta fase aproveitou-se algumas funções desenvolvidas anteriormente, diminuindo assim a duplicação de código. Desta forma, primeiramente foram desenvolvidas funções destinadas à geração de modelos, mas com o tempo, estas funções foram generalizadas e colocadas numa classe à parte servindo tanto para geração como para a selecção de modelos e ainda para a combinação do resultado final. As dificuldades já foram em menor número nesta fase, sendo preciso explorar ainda mais para o correcto desenvolvimento do código. Nesta fase a base de dados foi sem dúvidas menor do que na fase anterior pelo facto de só ser preciso acrescentar mais algumas informações principalmente sobre os *Ensembles*, já que muita informação utilizada aqui já tinha sido guardada na fase anterior.

A combinação do resultado final foi uma continuação de desenvolvimento de código, pois nesta fase já estava implementada a maioria do trabalho, sendo que as novidades não foram muitas. Em relação à base de dados as novidades também foram poucas, por se ter utilizado informação que já tinha sido guardada anteriormente e esta fase em si não exigia que se guardasse muita informação, não sendo precisas muitas tabelas. Portanto, na base de dados a novidade foi o aparecimento de uma função de combinação responsável por combinar os modelos e assim produzir o resultado final.

Seguindo a estratégia, foi mais fácil descobrir e resolver os problemas quando eles apareciam. Uma outra forma de tentar minimizar os problemas e melhor organizar o código foi o desenvolvimento de pequenas funções em vez de fazer o trabalho como um todo. Isso além de minimizar os problemas permitiu a não duplicação de código, pois em certos casos fez-se a generalização de alguns dados de forma a que as funções pudessem ser aproveitadas em mais do que uma fase.

5 Resultados experimentais

Neste capítulo são apresentados os resultados gerais obtidos pela execução da aplicação desenvolvida. Os testes descritos neste capítulo foram efectuados com *DataSets* retirados do repositório UCI Machine Learning [UCI11]. Um dos *DataSet* foi “machine.data”, com o seguinte domínio:

MYCT: continuous

MMIN: continuous

MMAX: continuous

CACH: continuous

CHMIN: continuous

CHMAX: continuous

class: continuous.

Para o setup experimental foi utilizada a validação cruzada através de uma função *10FoldCrossValidation* que faz a partição do conjunto inicial em dez partes.

Para a geração dos modelos foi feito o seguinte teste:

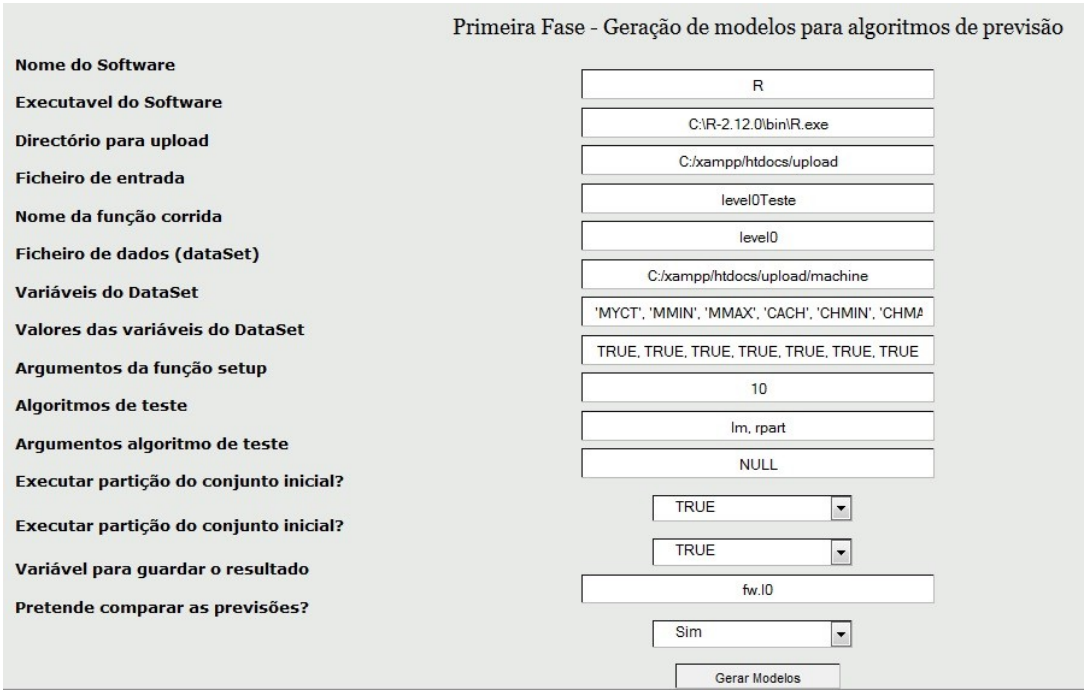

Figura 5.1: Exemplo de execução da geração de modelos

O nome do ficheiro de entrada é escrito sem a extensão, o PHP encarrega-se de a colocar. O nome do *DataSet* foi colocado indicando o caminho completo. Neste caso pretendia testar dois algoritmos de teste: o *lm* e o *rpart* que não têm parâmetros, pelo que o espaço correspondente aos parâmetros deve ser preenchido com NULL. Ao fornecer TRUE para as variáveis correspondentes à partição do conjunto inicial, a função principal chama a função de validação cruzada TenFoldCrossValidation que faz a partição do conjunto em 10 partes e a segunda variável que também possui o valor TRUE faz com que seja feita a partição em conjuntos de treino e teste. A variável para guardar o resultado final neste caso foi o "fw.l0" porque a função responsável por gerar os modelos guarda o resultado nesta variável. Foi escolhido fazer a comparação das previsões e neste caso foram guardadas informações a serem usadas na comparação. Com os dados passados, estes foram trabalhados e o comando passado ao R para executar a função level0 foi o seguinte:

fw.l0 <- level0('C:/xampp/htdocs/upload/machine', c('MYCT', 'MMIN', 'MMAX', 'CACH', 'CHMIN', 'CHMAX', 'class'), c(TRUE, TRUE, TRUE, TRUE, TRUE, TRUE, TRUE), c(10), c('lm', 'rpart') , NULL, TRUE, TRUE).

Ao carregar no botão "Gerar Modelos" que foi bem sucedido, foi apresentada a seguinte janela:

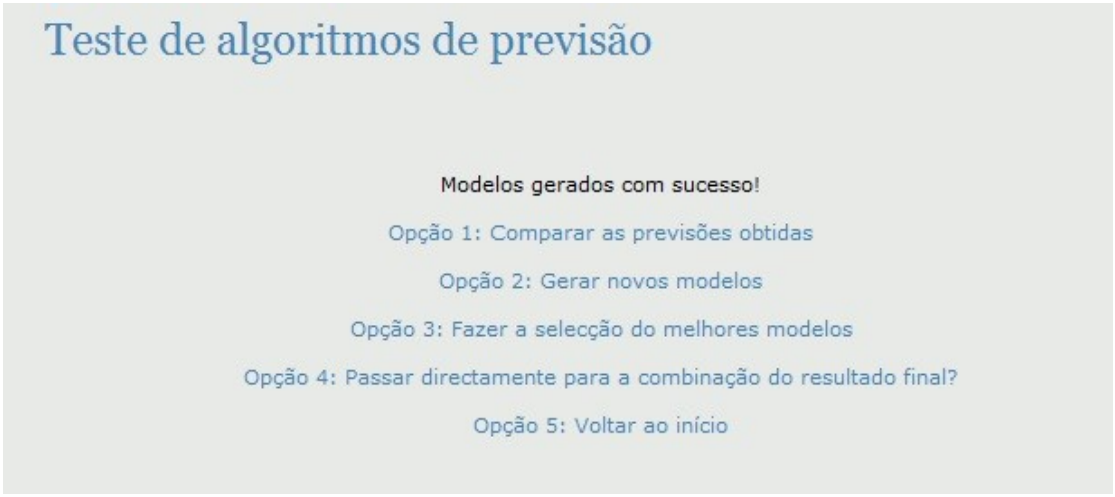

Figura 5.2: Caso de sucesso ao gerar modelos

Neste caso, os modelos foram gerados com sucesso e o resultado ficou guardado no directório para upload, especificado na janela anterior. Ao escolher a opção "Comparar as previsões obtidas", foi apresentada uma janela em que os dados fornecidos para efectuar a comparação são os apresentados na figura que se segue.

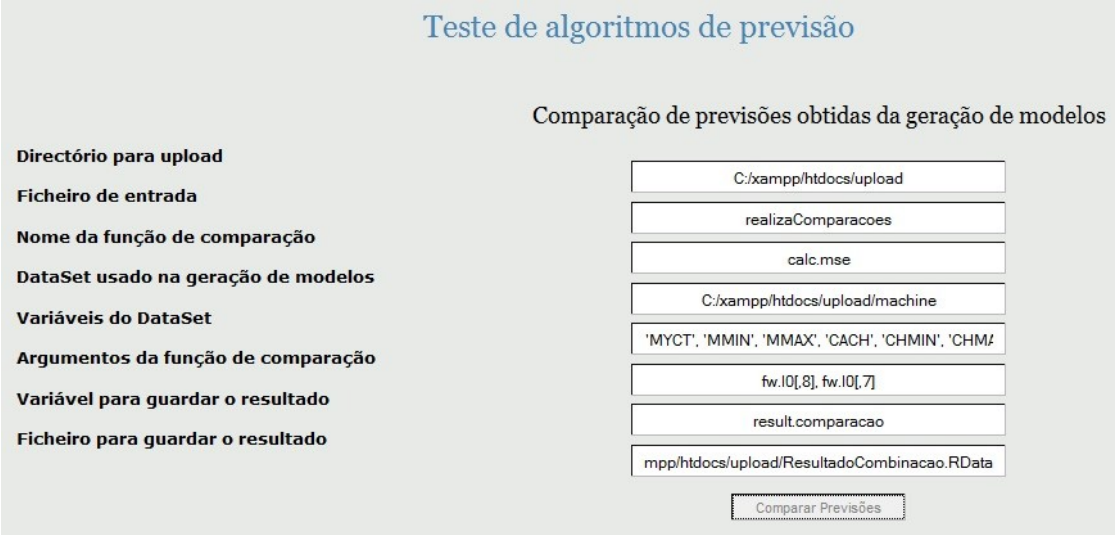

Figura 5.3: Comparação das previsões obtidas por um determinado modelo

A função utilizada para fazer a comparação é o "calc.mse" que faz a comparação segundo o erro médio quadrado. Esta comparação utiliza o mesmo *DataSet* utilizado na geração de modelos e consequentemente as variáveis são as mesmas. A função foi executada e o resultado ficou guardado na variável result.comparação, no ficheiro ResultadoComparação.RData no directório especificado que foi visualizado da seguinte forma.

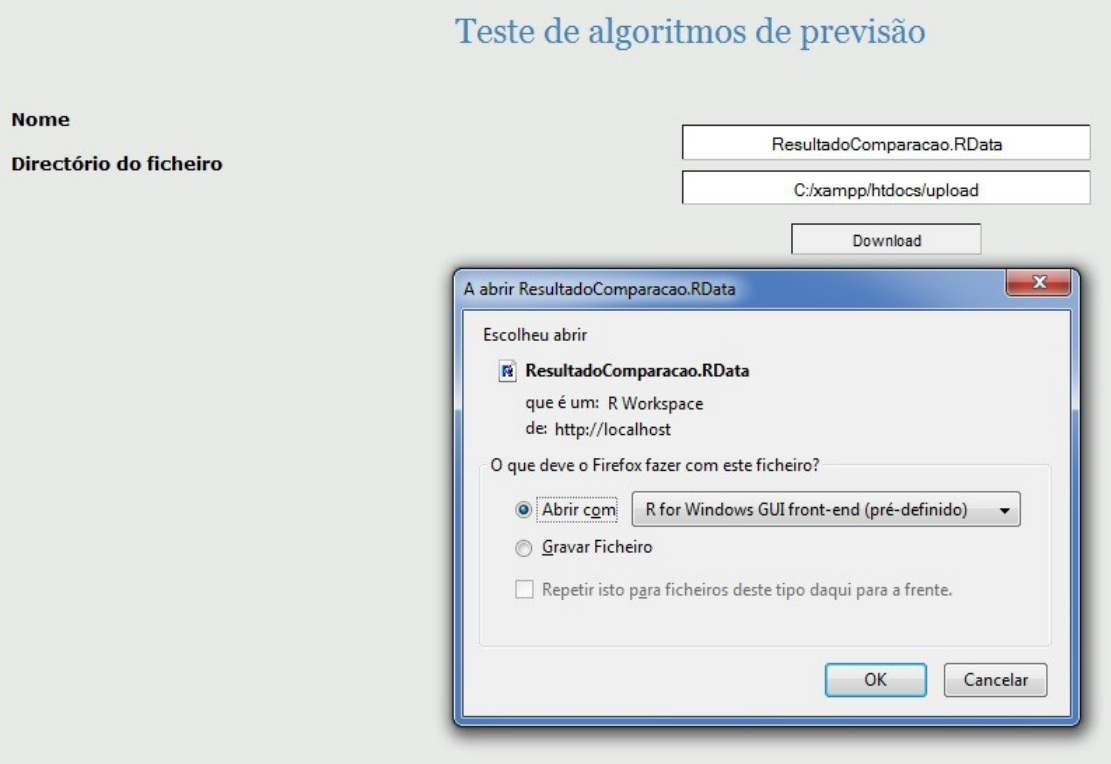

Figura 5.4: Ver resultado da comparação das previsões geradas pelos modelos

No diálogo acima foi escolhida a primeira opção pelo que o resultado é ilustrado na figura seguinte.

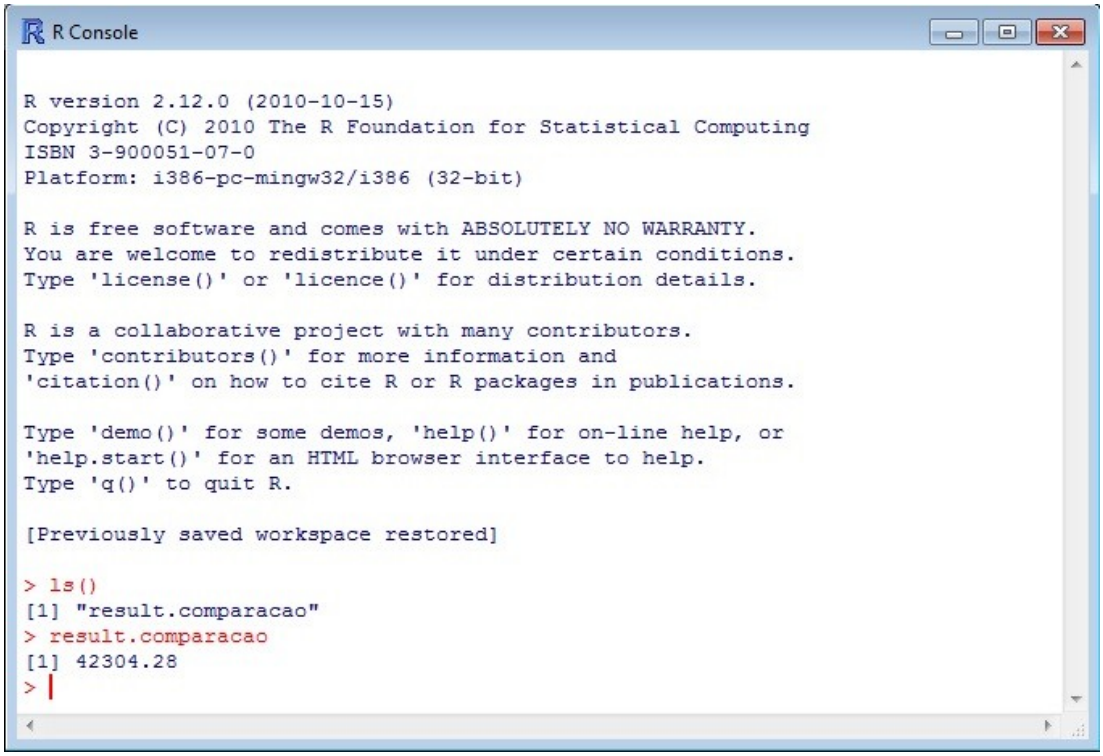

Figura 5.5: Resultado da comparação de previsões executada em R

O resultado corresponde à um valor numérico, tendo sido calculado através do erro médio quadrado.

Após verificar o resultado das previsões optou-se por fazer a selecção dos melhores modelos, cuja interface é ilustrada na figura que se segue:

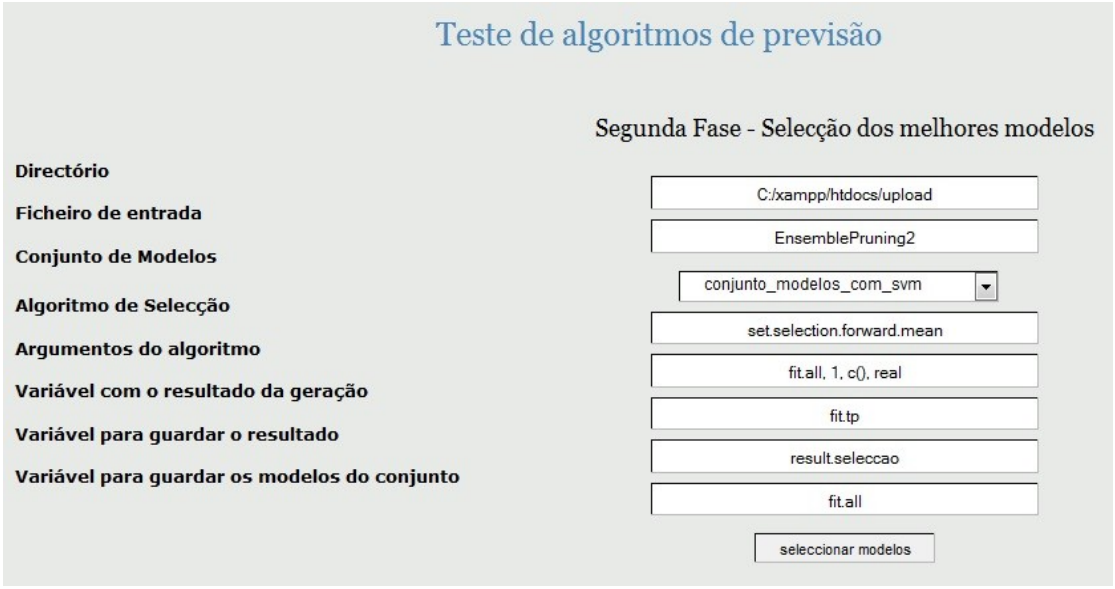

Figura 5.6: Selecção dos melhores modelos

Com os dados passados, a função “set.selection.forward.mean” foi executada e o resultado produzido foi uma lista em que cada elemento corresponde a um índice do modelo pertencente ao conjunto de modelos “conjunto_modelos_com_svm” que foi escolhido a partir de uma *combobox*. Esta *combobox* mostra todos os conjuntos de modelos produzidos até o momento e cabe ao utilizador escolher aquele que deseja utilizar para teste. Este conjunto, assim como os seus modelos constituintes, encontram-se guardados na base de dados do projecto. Para que a combinação pudesse usar os melhores modelos seleccionados, foi necessário desenvolver uma função em PHP que a partir dos índices retorna a designação dos respectivos modelos. Depois da execução da respectiva função, foi criado o “*ensemble*”, isto é o subconjunto constituído pelos modelos mencionados anteriormente. Este ficou guardado na base de dados para ser usado futuramente. A execução desta fase foi bem sucedida, o que quer dizer que os melhores modelos segundo a função “set.selection.forward.mean” foram seleccionados, apresentando a seguinte janela:

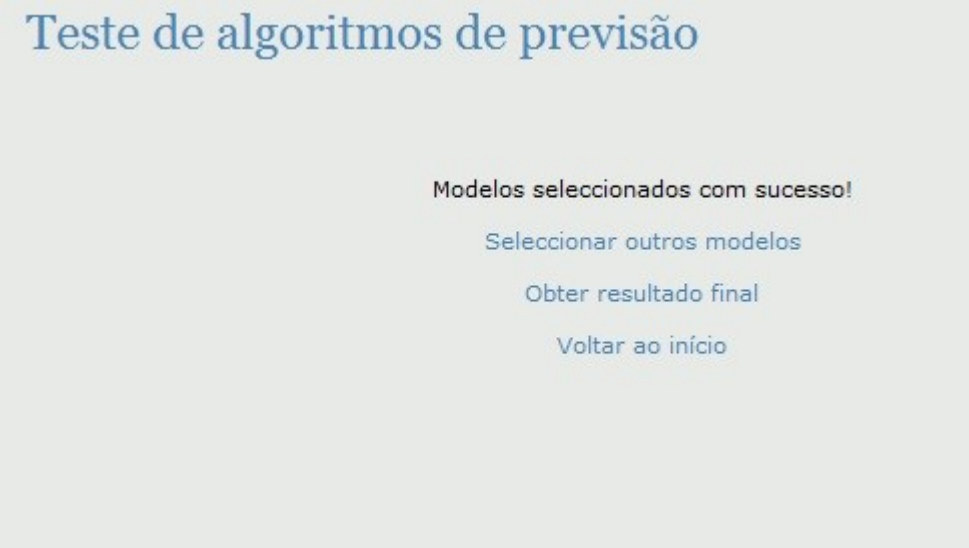

Figura 5.7: Sucesso ao seleccionar melhores modelos

Nesta janela foi escolhida a opção “Obter resultado final”, em que a respectiva página foi preenchida com os dados apresentados na figura seguinte.

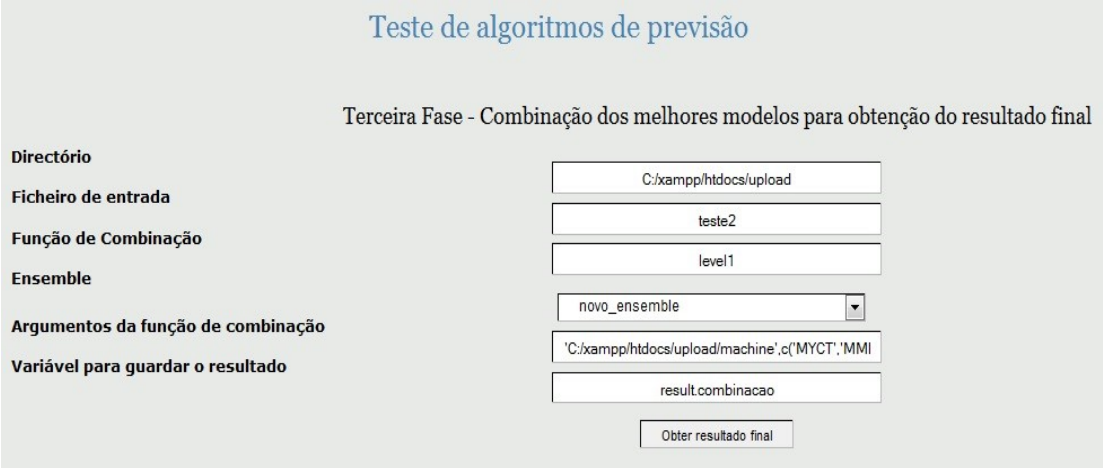

Figura 5.8: Combinação do resultado final, em que os modelos foram obtidos pela fase de selecção dos melhores modelos

À semelhança dos outros exemplos, foi pedido o directório e o ficheiro de entrada. Para esta fase destaca-se a função de combinação, responsável por fazer a combinação dos modelos pertencentes ao *ensemble* “novo_ensemble” que se encontra guardado na base de dados. Os argumentos desta função também foram passados pelo utilizador, assim como a variável onde se pretende guardar o resultado. Ao submeter estes dados, o resultado foi um caso de sucesso que é ilustrado na figura seguinte.

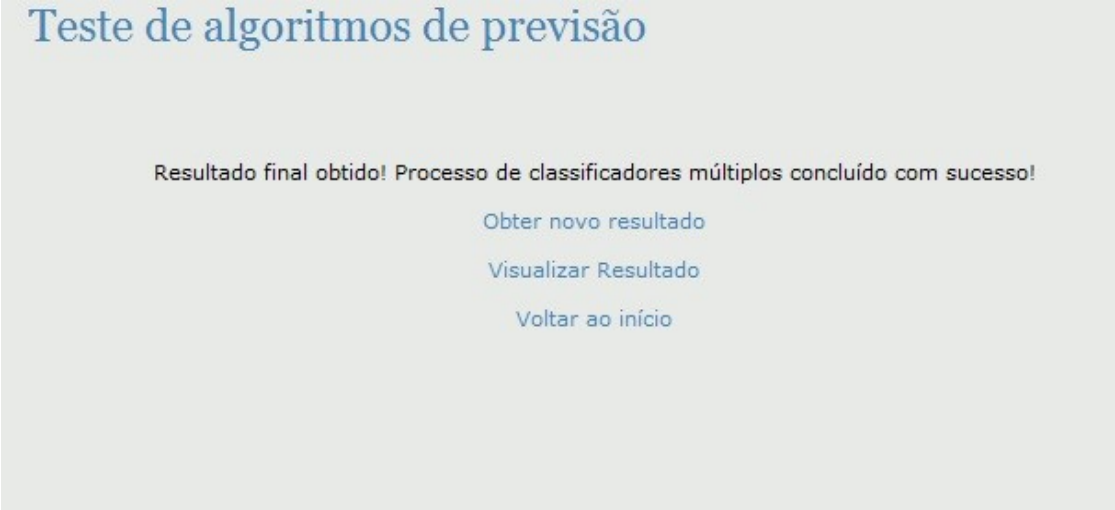

Figura 5.9: Sucesso ao combinar melhores modelos para obtenção do resultado final

Ao escolher a opção “Visualizar Resultado” foi apresentada uma janela ilustrada a seguir:

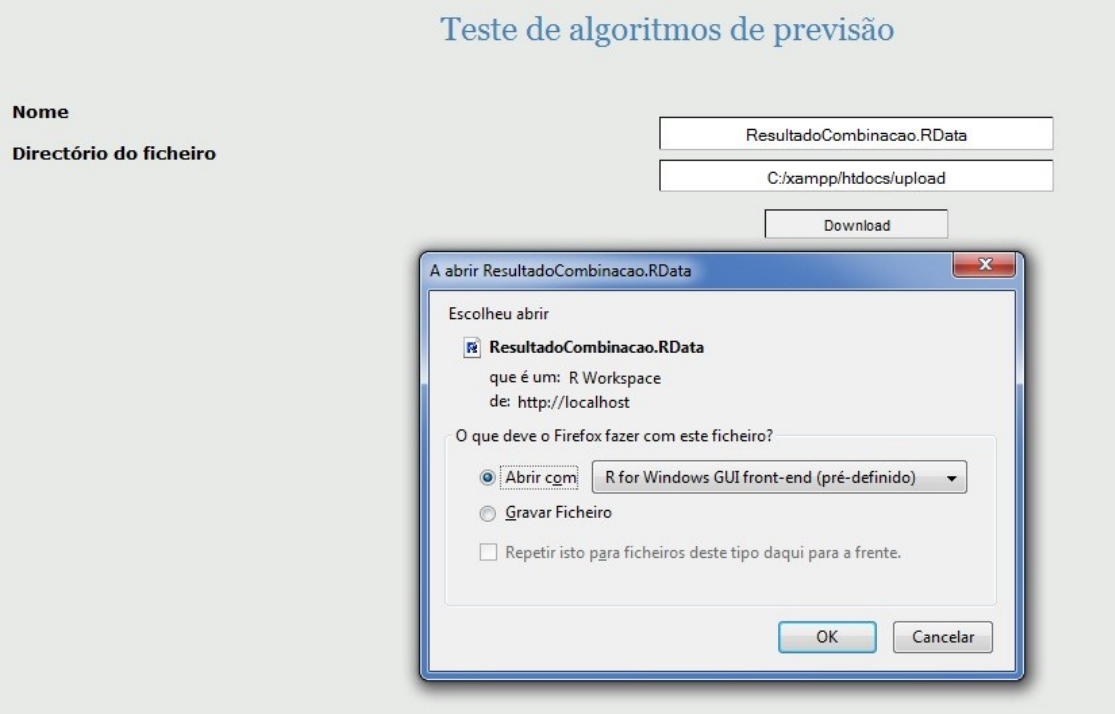

figura 5.10:Visualização do resultado final

Em que ao introduzir o nome do ficheiro contendo o resultado e o respectivo directório, foi apresentada uma janela de diálogo em que foi escolhida a opção “Abrir com” e o respectivo ficheiro foi aberto. Aqui terminou uma execução do processo de classificadores múltiplos.

Um outro exemplo de execução foi feito apenas com as fases de geração de modelos e combinação final, sendo que a diferença encontra-se na combinação que é feita de forma diferente. Sendo assim, quando a primeira fase é executada com sucesso, foi escolhida a opção

4 “Passar directamente para a combinação do resultado final”. Feita a escolha, apareceu uma nova janela correspondente a combinação do resultado final sem passar pela selecção que é ilustrada a seguir.

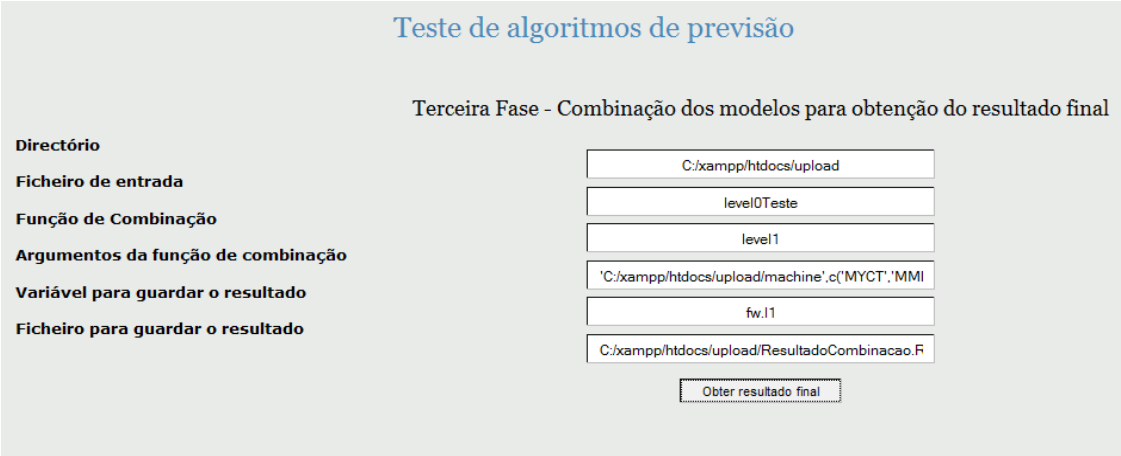

Figura 5.11: Combinação do resultado final sem passar pela fase de selecção dos melhores modelos

A diferença entre esta combinação e a outra é que nesta não foi pedido para escolher qual o *ensemble* que se pretendia utilizar. Posto isso, a função de combinação foi executada com os argumentos que lhe foram passados e o resultado foi guardado no ficheiro especificado. Esta fase foi executada com sucesso e à semelhança da combinação final feita a partir da selecção de modelos, foi possível ver o resultado. A figura seguinte mostra apenas parte do ficheiro contendo o resultado uma vez que se trata de um ficheiro com um elevado número de dados. Este ficheiro contém as variáveis do *DataSet*, o valor real, as previsões produzidas pelos algoritmos *lm* e *rpart* e a média das previsões.

	MYCT	MMIN	MMAX	CACH	CHMIN	CHMAX	target	lm	rpart	mean
1	125	256	6000	256	16	128	198	419.71347765	134.54054	277.127009
2	29	8000	32000	32	8	32	269	314.72043876	301.31250	308.016469
3	29	8000	32000	32	8	32	220	291.01790282	269.20000	280.108951
4	29	8000	32000	32	8	32	172	311.60296140	316.00000	313.801481
5	29	8000	16000	32	8	16	132	198.32391677	134.54054	166.432229
6	26	8000	32000	64	8	32	318	335.24241798	301.31250	318.277459
7	23	16000	32000	64	16	32	367	457.83835893	296.18750	377.012929
8	23	16000	32000	64	16	32	489	457.45578710	288.66667	373.061227
9	23	16000	64000	64	16	32	636	593.66794978	269.20000	431.433975
10	23	32000	64000	128	32	64	1144	874.00190686	475.66667	674.834287
11	400	1000	3000	0	1	2	38	-2.59223125	39.52381	18.465789
12	400	512	3500	4	1	6	40	-0.05607988	39.85039	19.897157
13	60	2000	8000	65	1	8	92	73.78073902	115.96774	94.874240
14	50	4000	16000	65	1	8	138	150.27988342	115.96774	133.123813
15	350	64	64	0	1	4	10	-33.42783778	41.12598	3.849073
16	200	512	16000	0	4	32	35	102.58250070	39.19841	70.890457
17	167	524	2000	8	4	15	19	5.37877853	40.45312	22.915952
18	143	512	5000	0	7	32	28	37.25278900	40.28906	38.770926
19	143	1000	2000	0	5	16	31	10.00067356	40.45312	25.226899
20	110	5000	5000	142	8	64	120	247.38936101	210.12500	228.757181
21	143	1500	6300	0	5	32	30	52.43479050	39.85039	46.142592
22	143	3100	6200	0	5	20	33	63.22625632	38.07087	50.648561
23	143	2300	6200	0	6	64	61	117.68615600	38.81395	78.250055
24	110	3100	6200	0	6	64	76	108.66739490	40.45312	74.560260
25	320	128	6000	0	1	12	23	11.40177241	38.62400	25.012886
26	320	512	2000	4	1	3	69	-18.00209969	39.19841	10.598157
27	320	256	6000	0	1	6	33	5.86545643	40.28906	23.077259
28	320	256	3000	4	1	3	27	-13.94210497	40.28906	13.173479
29	320	512	5000	4	1	5	77	2.93175907	39.19841	21.065086
30	320	256	5000	4	1	6	27	5.43401974	38.07087	21.752443
31	25	1310	2620	131	12	24	274	82.84331317	132.16216	107.502738
32	25	1310	2620	131	12	24	368	74.67044367	125.94737	100.308906
33	50	2620	10480	30	12	24	32	94.25911446	133.94444	114.101779
34	50	2620	10480	30	12	24	63	98.17043588	130.97143	114.570932

Figura 5.12: Parte do resultado final do processo de classificadores múltiplos.

Nos exemplos apresentados foram evidenciadas as partes mais importantes do trabalho com destaque para as três fases constituintes do processo, uma vez que a demonstração completa do trabalho realizado ocuparia muito espaço desta tese. Algumas das opções que não foram mostradas são:

- Voltar ao início – esta opção pode ser escolhida em qualquer fase do processo e permite voltar à pagina inicial;
- Erro – acontece quando uma acção escolhida pelo utilizador não acontece conforme o previsto. Ao ser apresentada a janela de erro, o utilizador pode optar por repetir a acção ou pode voltar ao inicio;
- Upload de ficheiro – esta opção permite fazer upload de ficheiros, que serão guardados num directório especificado pelo utilizador. Isso porque o utilizador pode querer ter todos os ficheiros que pretende utilizar no mesmo directório;

- Repetir a última fase executada – por uma razão qualquer, o utilizador pode querer repetir a execução. Por exemplo, depois de executar uma determinada fase é que se lembrou que utilizou um determinado valor para um parâmetro e não outro qualquer.

Fazendo uma análise dos resultados obtidos, estes são satisfatórios, embora não se possa dizer que executar o processo com as três fases seja melhor do que executá-lo apenas com a geração de modelos e a combinação final ou vice-versa, porque isso depende muito dos dados usados em cada fase, como por exemplo: a quantidade de dados utilizados, assim como o número de modelos gerados, etc.

6 Conclusões e Trabalho Futuro

Ao longo desta Dissertação intitulada “Protótipo de testes para aprendizagem com classificadores múltiplos” foi desenvolvida uma aplicação web que permite testar algoritmos de previsão. Apresenta-se como uma alternativa inteligente à utilização da linguagem R em sistemas de classificadores múltiplos, passando por três fases diferentes em que primeiramente são executados vários algoritmos, com várias combinações de parâmetros que produzem modelos. A seguir são aplicados alguns critérios de forma a escolher os melhores modelos produzidos anteriormente, embora esta fase não seja obrigatória. Na terceira e última fase as previsões produzidas pelos modelos são combinadas produzindo um resultado único para o todo o processo. Os classificadores múltiplos apresentam bons resultados quando comparados com classificadores individuais e estes bons resultados são possíveis quando existe diversidade. Esta diversidade pode ser conseguida através da utilização de algoritmos de teste diferentes porque desta forma, os comportamentos são diferentes, produzindo erros diferentes em diferentes situações o que leva a que o resultado final produza um erro menor através de uma boa estratégia de combinação.

A realização desta Dissertação foi um desafio uma vez que o tema classificadores múltiplos era desconhecido pelo autor. Ao longo desta dissertação foi feito um estudo intenso sobre os classificadores múltiplos de modo a compreender o seu funcionamento, o porque da sua utilização, as suas vantagens quando comparados com classificadores individuais. Desta forma, depois de fazer um estudo sobre o tema, foi preciso traçar uma estratégia que fosse a melhor possível para a resolução do problema, de forma a cumprir os objectivos propostos. Surgiram algumas dificuldades durante a implementação deste trabalho, mas que foram superadas.

Em relação às ferramentas utilizadas, constatou-se o PHP é uma boa ferramenta para desenvolvimento de aplicações web, assim como a linguagem R mostra-se uma boa ferramenta para cálculos, apresentando ambas várias funcionalidades importantes para o respectivo fim. E, sendo que é possível fazer a interacção entre as duas linguagens, a utilização das duas mostrou-se uma boa escolha para a implementação da aplicação desenvolvida. Além dos objectivos

propostos, o trabalho desenvolvido permitiu constatar a viabilidade da integração da linguagem R com tecnologias web como por exemplo: PHP, HTML, MySQL, CSS e JavaScript.

6.1 Satisfação dos Objectivos

Os objectivos definidos inicialmente foram cumpridos. Depois de elaborado um estudo sobre os classificadores múltiplos, foi possível implementar uma solução baseada no desenvolvimento de uma aplicação web, onde é possível utilizar a aplicação para testar algoritmos de previsão, sendo que nesta aplicação deu-se maior destaque aos *ensembles* heterogéneos de forma a garantir uma maior diversidade. O número de parâmetros utilizados é flexível, dependendo dos algoritmos que se pretende utilizar. Desta forma é possível testar não apenas um ou alguns algoritmos específicos uma única vez, mas testar vários algoritmos um número de vezes variável, dependendo dos argumentos passados a cada algoritmo. É uma aplicação flexível, que permite a utilização não de uma determinada função de combinação ou de selecção mas a utilização de funções variadas para este fim. Ainda a aplicação contém uma particularidade importante que é a seguinte: para além de permitir a utilização de classificadores múltiplos passando pelas três fases: geração de modelos, selecção dos melhores modelos e combinação do resultado final, permite a sua utilização apenas com duas fases: geração de modelos e combinação do resultado final, conforme definido por alguns autores.

A aplicação desenvolvida permite a utilização dos classificadores múltiplos de uma forma mais agradável, flexível e de fácil utilização facilitando a componente de teste dos algoritmos de previsão, uma vez que o processo experimental é complexo e esta complexidade aumenta cada vez que aumentam os dados a serem utilizados nos testes. Sendo assim, o utilizador não precisa ficar o tempo todo à espera para ver a execução sendo simplesmente necessário fornecer os dados a serem utilizados e o processo será automático, processo este que pode levar muito tempo, dependendo da quantidade de dados a utilizar. Além disso, pode ser utilizado não só por especialistas que queiram testar o seu próprio código para as diferentes fases do processo, mas também por utilizadores comuns que utilizem código já desenvolvido.

6.2 Trabalho Futuro

Os trabalhos e pesquisas no campo dos classificadores múltiplos têm vindo a aumentar, embora ainda haja muito trabalho para fazer, tanto a nível teórico como a nível experimental.

No seguimento deste trabalho destacam-se alguns pontos como:

- alargar o trabalho no sentido de aceitar código escrito não apenas em R, mas em outras linguagens de programação para cálculos. Neste contexto tornar a aplicação mais flexível de forma que, para além de executar algoritmos escritos em linguagens diferentes, possa executá-los no mesmo processo de classificadores múltiplos;
- integrar o trabalho de forma a poder ser executado pelas máquinas de *cluster* existentes na FEUP, de forma a diminuir o tempo de execução. Dado que os algoritmos são independentes uns dos outros, a sua execução através de *clusters* da FEUP seria óptimo

porque iria diminuir consideravelmente o seu tempo de execução. Desta forma, esta integração seria possível através da criação de scripts que seriam automatizados uma vez que os parâmetros variam. Esta execução seria feita via *ssh*(Secure Shel). Para a utilização deste serviço disponibilizado pela FEUP, é necessário um procedimento de pedido de autorização em que caso seja aceite, é atribuído um *username* e uma *password* que permitirão o utilizador utilizar o serviço [Grid05].

Referências

[Pol06] Robi Polikar. Ensemble Based in Decision Making, 2006.

[Mor08] João Mendes Moreira. Travel time prediction for the planning of mass transit companies: a machine learning approach, Novembro 2008.

[PHP10] Página oficial do PHP, 2010. http://php.net/ (acedido a última vez em Dezembro de 2010).

[R10] Página oficial da linguagem R, 2010. http://www.r-project.org/ (acedido a última vez em Dezembro de 2010).

[Wol92] David H. Wolpert. Stacked Generalization, 1992.

[Mer96] Christopher J. Merz. Dynamical Selection of Learning Algoritms, 1996.

[Mor01] J. Mendes Moreira, Machine Learning and Data Mining in Pattern Recognition, Dynamic Selection, 2001.

[Kun04] Ludmila I. Kuncheva. Combining Pattern Classifiers, Methods and Algorithms, 2004.

[Sew07] Martin Sewell. Ensemble Learning, Abril 2007.

[PP07] Devi Parikh and Robi Polikar. An Ensemble-Based Incremental Learning Approach to Data Fusion, Abril 2007.

[SAS10] SAS Enterprise Miner, 2010. http://www.sas.com/technologies/analytics/datamining/miner/ (acedido a última vez em Dezembro de 2010).

[SPSS10] SPSS Modeler, 2010.http://www.spss.com/software/modeling/modeler-pro/ (acedido a última vez em Dezembro de 2010).

[RM10] RapidMiner, 2010. http://rapid-i.com/content/view/181/190/ (acedido última vez em Dezembro de 2010).

[WEKA10] WEKA, 2010. http://www.cs.waikato.ac.nz/ml/weka/index.html (acedido última vez em Dezembro de 2010).

[NYT09] The New York Times , 2009. http://www.nytimes.com/2009/01/07/technology/business-computing/07program.html?_r=1 (acedido última vez em Dezembro de 2010).

[PHTML10] PHP Faq, 2010. http://pt2.php.net/manual/pt_BR/faq.html.php (acedido última vez em Dezembro de 2010).

[PHPDB10] PHP Faq, 2010. http://pt2.php.net/manual/pt_BR/faq.databases.php (acedido última vez em Dezembro de 2010).

[FA05] John Fox and Robert Anderson. Using the R statistical compuring environment to teach social statistics courses, Janeiro 2005.

[Tor06] Luís Torgo. Introdução à Programação em R, Outubro 2006.

[MySQL11] Pagina oficial do MySQL, 2011. http://www.mysql.com/ (acedido última vez em Janeiro de 2011.

[WMySQL11]Why MySQL, 2011. http://www.mysql.com/why-mysql/ (acedido última vez em Janeiro de 2011).

[DocMySQL11]Documentation MySQL, 2011. http://dev.mysql.com/doc/refman/4.1/pt/features.html (acedido última vez em Janeiro de 2011).

[Grid05] Princípios orientadores de utilização e gestão. Documento interno, Faculdade de Engneharia da Universidade do Porto. Disponível em http://grid.fe.up.pt/files/PrincipiosUtilizacaoCluster.pdf, Setembro de 2005.

[WCTS07] Joerg D.Wichard, Henning Cammann, Thomas Tolxdolff, Carsten Stephan. Early Detection of Prostate Cancer with Classifier Ensembles, 2007

[UCI11] UCI Machine Learning Repository, 2011. http://archive.ics.uci.edu/ml/datasets.html (acedido última vez em Janeiro de 2011).

[RPAT04] Niall Rooney, David Patterson, Sarah Anand and Alexey Tsymbal. Dynamic Integration of Regression Models. In *International Worshop on Multiple Classifier Systems*, volume LNCS 3077, pags 164-173. Springer, 2004.

Anexo A: PHP e MySQL

A.1 Ligação do PHP com o servidor da base de dados MySQL

Especificação da classe MDB2

```
define('MDB2_DIR', 'C:/xampp/php/PEAR/');
```

Ligação à base de dados, definidos no ficheiro startup.php:

```
require_once(MDB2_DIR.'MDB2.php');
$mdb2 =& MDB2::factory('mysql://root:@localhost/geracao_modelos');
if (PEAR::isError($mdb2))
{
    die($mdb2->getMessage().' - '.$mdb2->getUserinfo());
}
$mdb2->setFetchMode(MDB2_FETCHMODE_ASSOC);
```

em que:

tipo de base de dados = “mysql”;

user = “root”;

password = “”; //neste caso não tem password

hostname = “localhost”;

nome da base de dados = “geracao_modelos”;

A.2 Bibliotecas

Especificação do caminho para as bibliotecas PEAR e Smarty utilizadas. Esta definição encontra-se no ficheiro php.ini:

include_path = ".;C:\xampp\php\PEAR\;C:\xampp\smarty\libs\"

A.3 Smarty

Definição e inclusão do smarty no ficheiro startup.php

```
define('SMARTY_DIR', 'C:/xampp/smarty/libs/');
require_once(SMARTY_DIR.'Smarty.class.php');
$smarty =new Smarty();
```

Anexo B: PHP e R

B.1 Ligação entre o PHP e o R

Instrução para executar um comando externo ao PHP:
string **exec** (string $command [, array &$output [, int &$return_var]])
em que:
$command é o comando a ser executado,
$output - caso esteja presente, é a saída produzida pela execução do comando,
$return_var - caso esteja presente, é a variável para onde será escrito o estado de retorno do comando executado.

Instrução para executar um comando do R via PHP:
exec($command, $result, $error);
em que:
$command = $path." --no-restore --no-save < $input_file > $output_file 2>&1";
onde:
$path é o caminho do executável do R,
$input_file é o ficheiro de entrada,
$output_file é o ficheiro de saída,
"--no-restore --no-save <" , ">" e "2>&1" são opções que devem ser adicionadas para o correcto funcionamento da execução.

$error representa o estado de retorno de execução do comando que pode ser de erro ou sucesso.

Instrução para gravar um ficheiro em formato RData no R:
save(var_result, file = 'file.RData')

em que var_result é a variável onde foi guardado o resultado que se pretende guardar no ficheiro e file é o ficheiro para onde se pretende guardar os dados.